TRSTimes

Volume 6. No. 2. - Mar/Apr 1993 - $4.00

ANOTHER ALL-NIGHT DEBUGGING SESSION AT TRSTIMES

LITTLE ORPHAN EIGHTY

Over the last couple of months, several of our new subscribers have expressed an interest in a program called BOOT5/CMD. This little gem was written by Adam Rubin, and allows almost instant change from Model 4 LS-DOS to Model 3 mode and LDOS. The problem with BOOT5/CMD is that it was written explicitly for LDOS 5.3.0. When LDOS was upgraded from 5.3.0 to 5.3.1, BOOT5/CMD refused to work on the later version of DOS.

In our Jan/Feb 1992 issue of TRSTimes, Roy Beck wrote an article called 'Boot your Hard Drive in LS-DOS and LDOS without a floppy!', to which I contributed a series of eight one-byte patches which would make BOOT5/CMD work flawlessly on LDOS 5.3.1.

In the documentation to BOOT5/CMD, Adam Rubin explicitly requests that his program be distributed in its original unmodified form. As a programmer myself, I certainly understand and respect Adam's request. Nothing is worse that sitting night after night, sweating over Basic or Assembly Language code, finally getting it the way you want it, only to have some bozo modify it to do things you might not want it to do, and then spread it around with your name all over it, leaving you to take the blame if the mods screw things up. Believe me, I understand. I have found a few of my own programs altered by other people, some even removing the copyright notices.

Anyway, after publishing the patches, we wanted to include the updated version on the next TRSTimes on Disk. Roy Beck wrote two letters to Adam, requesting permission to distribute the modified version. In the correspondence, he included a printout of the patches and a disk of the modified program. We never received an answer and, thus, it is rather obvious that we do not have specific permission to share the 5.3.1 version of BOOT5/CMD with the readers.

Therefore, for those of you who requested a copy of the new BOOT5/CMD file and were turned down, I just want you to know that we were not trying to be mean. We simply wanted to play by the rules.

Now, not having the permission to distribute the actual modified file does not mean that I cannot give you the information on how to make your LDOS 5.3.0 version of BOOT5/CMD work on 5.3:1. Below is a reprint of the patches we published in the Jan/Feb 1992 issue.

PATCH BOOT5/CMD (D01,92=6A:F01,92=62)

PATCH BOOT5/CMD (D03,5F=31:F03,5F=30)

PATCH BOOT5/CMD (D04,2B=31:F04,2B=30)

PATCH BOOT5/CMD (D04,D3=D9:F04,D3=F6)

PATCH BOOT5/CMD (D04,D9=50:F04,D9=6D)

PATCH BOOT5/CMD (D04,DC=51:F04,DC=6E)

PATCH BOOT5/CMD (D04,E1=D3:F04,E1=F0)

PATCH BOOT5/CMD (D04,E4=12:F04,E4=2F)

Install the patches, but do have respect for the author's wishes. Do not distribute the modified program. If someone else wants the version that works on 5.3.1, give them a copy of the patches and let them modify their own copy.

Changing the subject, let me divert your attention to the MISOSYS ad elsewhere in this issue. This is undoubtably the best sale of TRS-80 software you will ever find. All the things you always wanted, but just never got around to buy, are there. And the prices are right. Just imagine, THE SOURCE 3-volume set for $10.00. They used to cost $99 each. Or, how about Leo Christopherson's Game Disk for $10.00. Do take a hard look at this ad and order the things you want now. Do not procrastinate as these are closeouts. When they are gone, there ain't no mo'.

I have already ordered (and recieved) THE SOURCE and PRO HARTFORTH. I already had a copy of THE SOURCE, which was obtained through the purchase of TRS-80 equipment locally, but it was time to get a fresh copy. This is also a reminder to our readers who have some of these programs, but 'have misplaced the manual'. Now is the time to BUY the programs you have enjoyed for years and, at the same time, make programming for the TRS-80 somewhat worthwhile for Roy Soltoff. I have stated this before, but it certainly is worth repeating - *we cannot afford to have Roy Soltoff leave the TRS-80 world entirely*. He is the last of the major leaguers. Oh sure, there are a few of us left still writing things for the machines, but none of us are up to his skills. Triple-A, maybe, but not the majors.

This issue of TRSTimes seems to focus heavily on the Model 4. Why don't we publish more material for the Models I and 3? Simply because you, the readers, are not submitting material for those machines. If you want new programs you must start the ball rolling by sending us your material. As you well know, the purpose of this publication is for all of us to share information. It was never my intention to dazzle (bore) the readers by writing each page myself. So get off you ol' rusty-dusty and share some of your experiences with the rest of us in TRS-80 land.

To Roy Beck, Chris Fara, Jim King, Henry Herrdegen and Frank Gottschalk, many thanks for helping to make this issue possible.

And now.....Welcome to TRSTimes 6.2.

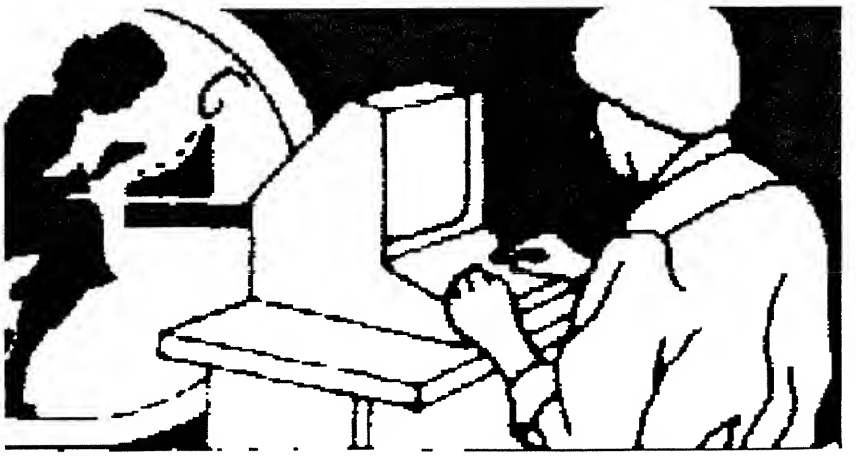
TRSTimes magazine

Volume 6. No. 2 - Mar/Apr 1993

PUBLISHER-EDITOR
Lance Wolstrup
CONTRIBUTING EDITORS
Roy Beck
Dr. Allen Jacobs
Dr. Michael W. Ecker
TECHNICAL ASSISTANCE
San Gabriel Tandy Users Group
Valley TRS-80 Users Group
Valley Hackers' TRS-80 Users Group

TRSTimes magazine is published bi-monthly by TRSTimes Publications. 5721 Topanga Canyon Blvd, Suite 4, Woodland Hills, CA. 91364. (818) 716-7154.
Publication months are January, March, May, July, September and November.

1993 subscription rates (6 issues):
UNITED STATES & CANADA: $20.00 (U.S. currency)
EUROPE, CENTRAL & SOUTH AMERICA: $24.00 for surface mail or $31.00 for air mail.
(U.S. currency only)
ASIA, AUSTRALIA & NEW ZEALAND: $26.00 for surface mail or $34.00 for air mail.
(U.S. currency only)
Article submissions from our readers are welcomed and encouraged. Anything pertaining to the TRS-80 will be evaluated for possible publication. Please send hardcopy and, if at all possible, a disk with the material saved in ASCII format. Any disk format is acceptable, but please note on label which format is used.



GEO4

a Model 4 educational game

(with lots of programming tricks)

by Lance Wolstrup

Steven, my youngest son, is now 12 years old, and we spend a lot of time together. I drive him to and from school; I drive him to soccer and baseball; I take him to ice-hockey games, the movies, the mall, and McDonalds. Living in Southern California, this means spending a lot of time in the car, going from one place to the next. As I have described in earlier issues of TRSTimes, we often play word games to pass the commute-time and, per Steven's request, I have turned our latest folly into a computer game. We call the game GEO4, because it is about geography and it is played on a Model 4.

As a game, GEO4 is not unique. I am sure that many people have at one time or another played a variation of it, but this version is totally fresh; that is, it is not a rewrite of something I have seen on some other machine; rather, it is the computer adaptation of the way Steven and I play the game - with some bells and whistles thrown in for good measure.

The original game was played to strengthen Steven's knowledge of U.S. geography, and I would begin by saying the name of state or a U.S. city. Steven would then use the ending letter of my city or state as the first letter of a new city or state. I would then respond with a city or state that began with the ending letter of Steve's entry. For example, I might start with MICHIGAN. Steven would then have to find a city or state beginning with N. Let's assume that he then says NANTUCKET. I would respond with TAMPA, and he would counter with ALABAMA, after which I would say ALAMAGORDO, etc.

The computer version works much the same. The major exception is that Steven (or the player) is no longer pitted against ol' Dad, but instead plays against a much worthier opponent, the Model 4. The other exception is that, before becoming the opponent, the Model 4 picks the first city or state randomly. This is done so that each game will begin differently. The player then begins the game by using the last letter of the computer pick to form his/her city or state.

Be advised that the Model 4 is indeed a worthy opponent. As a matter of fact, the reason the player always goes first is that otherwise there would be no chance of ever winning. The best that could be hoped for was a tie. However, since the player is always first, he/she can win, but you have to have a good knowledge of U.S. geography, while using a good game strategy. But then, that's the whole idea, isn't it?

HOW TO PLAY GEO4

GEO4/BAS begins by displaying its name in large letters at the center of the screen while the program is busy taking care of internal housekeeping. Almost immediately, however, the screen changes and the player is asked to type his/her name.

The player can respond to the name prompt by pressing <SHIFT><UP-ARROW> to end the game, or typing a name and then pressing the <ENTER> key.

Assuming that the player typed his/her name and pressed <ENTER>, the scoreboard is displayed at the top of the screen and the player is prompted to press the <ENTER> key when ready to begin the game.

If the player is not satisfied with the name, pressing <SHIFT><UP-ARROW> will return to the name prompt, otherwise pressing <ENTER> will begin the game.

At this point the computer selects a city or state at random and the game then begins by letting the player go first. The player must, however, base his/her selection on the last letter of the randomly picked starting city or state.

When the player responds to this, or future selections, basically five things can happen.

1. If the player cannot think of a city or state, he/she can press <SHIFT><UP-ARROW>. This will allow the TRS-80 to go again, and if it can find a city or state beginning with the last letter of the previous city or state, it wins the game. If not, the game ends in a tie.

2. The player types a city or state that does not begin with the last letter of the previous city or state. If this happens, GEO4 reponds with an error message telling the player that his/her selection must begin with whatever letter the previous enty ended with. The player is then allowed to try again.

3. The player types in a city or state that is not included in the database. This will most often happen if the player misspells the selection. An error message will inform the player that the selection was not found in the database, and the player is then allowed to try again.

4. The player types in a city or state that has already been used. Since this is not allowed, GEO4 displays an error message telling the player that this particular city or state has already been used. The player is the allowed to try again.

5. The player types in a valid city or state.

If the types in a valid city or state, the TRS-80 takes over and selects a city or state beginning with the last letter of the player selection. If this is not possible, the player wins the game, otherwise the game continues with the player typing in the next city or state. This goes on until either the computer cannot come up with a valid response, or the player gives up by pressing <SHIFT><UP-ARROW>.

Ending the game, displays the name of the winner, and the player is prompted if he/she would like to play again. Responding with <N> or <SHIFT><UP-ARROW> ends the game. The screen is erased and control is passed back to Basic's Ready prompt.

Pressing <Y> to the 'play again' prompt returns the player to the 'press <ENTER> to begin' prompt.

ABOUT THE CODE

As the subtitle of this article suggests, I used several 'programming tricks' to accomplish the look and feel of GEO4. First and foremost, the screen is split into three individual screens, allowing the top and bottom portions of the screen to remain stationary, while the center portion is free to scroll normally. I have written about this feature in previous issues of TRSTimes, so I'll just mention that the code to isolate the bottom of the screen is found in the subroutine in lines 50 - 59. Enter this subroutine with variable R = the number of lines you wish the normal top screen to be. For example, R=19 makes DOS think that the screen is only 19 rows, thus bottom protecting 5 lines.

After the scoreboard is displayed, line 105 scroll-protects 6 lines from the top of the screen by setting bits 1 and 2 of memory location B94H. This has also been talked about in earlier issues of this magazine.

Since the program changes important pointers in DOS, line 10 disables the <BREAK> key by setting bit 4 of memory location 7CH. Disabling the <BREAK> key in this manner is many, many times faster than using the SYSTEM command. It is done so the only way to exit the program is through line 120, where all changes are restored to their normal values.

GEO4 will only allow the user to answer prompts in uppercase. No matter how many times you press the <CAPS> key, all keystrokes will be uppercase. Line 31 takes care of that by constantly setting bit 5 of memory location 74H.

Lines 1000 - 1072 contain the data for the game. There are currently 435 pieces of data, and this is indicated by variable D in line 10. If you wish to enter additional data, make absolutely sure that each piece of data is placed so that all data remain in alphabetical order, then change the value in variable D to reflect the new number.

With this many pieces of data, how does GEO4 manage to find individual states or cities with such speed? Because the data is stored in alphabetical order, the program can utilize a search technique called a binary search. If you've ever played the 'guess a number between 1-100' game, you've probably used the technique without realizing it.

As a demonstration, let's play the above-mentioned game, and let's make the number we need to guess 5. The key to the game is to constantly eliminate half of the numbers until we reach the correct number.

1st guess: 50 <--- Lower

Since we now know that is less than 50, we can throw out all the numbers from 50 - 100. The game has now changed to 'guess a number between 1 and 49'.

2nd guess: 25 <--- Lower
Now we can throw out all the numbers from 25 to 49. The range is now 1 to 24.

3rd guess: 12 <--- Lower
Throw out numbers from 12 to 24. The new range is 1 to 11.

4th guess: 6 <--- Lower
Throw out numbers from 6 to 11. The range is now 1 to 5

5th guess: 3 <--- Higher
Throw out the numbers from 1 to 3, leaving us the range of 4 to 5.

6th guess: 4 <--- Higher
The answer is now obvious.

7th guess: 5 <--- Correct

As should be easy to see from the example, each guess cuts the range in half. To be more precise, we subtract the low range from the high range, divide the result by two, keep just the integer portion, and then add the low range. Confusing, isn't it?

For example, our first guess was derived in this manner:

100 - 1	= 99	(high range minus low range)
99 / 2	= 49.5	(above result divided by 2)
INT(49.5)	= 49	(keep the integer portion)
49 + 1	= 50	(add low range)

You might wonder why I bother to do these calculations when it is so obvious that the halfway point between 1 and 100 is 50.

The reason, of course, is that we need to define a formula that will hold true always, even when we need to find the halfway point between 4 and 5, as in our 6th guess above.

This algorithm, found in lines 76 (and 313) of the GEO4/BAS listing, reads:

M=INT((R-L)/2)+L

If we list the '1 - 100' game up step-by-step, we would show the low range as the left-most number (L), the halfway number in the middle (M) and the high range as the right-most number (R).

gUess	L	M	R
(1)	1	50	100
(2)	1	25	49
(3)	1	12	24
(4)	1	6	11
(5)	1	3	5
(6)	4	4	5
(7)	5	5	5

The algorithm holds true, finding M in guess 1. Since the number we are looking for is smaller than 50, we now need to adjust the parameters:

L remains the same.
R becomes M-1
M is calculated as INT((R-L)/2+L

This continues through guess 4. However, when we arrive at guess 5, we are told that the number is now higher. We adjust the parameters accordingly:

L becomes M+1
R remains the same
M is calculated as INT((R-L)/2)=L

This code can be found in lines 74 (and 311) of the listing to GEO4/BAS.

The entire subroutine in lines 70 - 79 handles the binary search for the users picks. The routine in lines 300 to 320 takes care of the binary search of the computer picks. In lines 71, variable U is compared to variable MX. If U is larger than MX then a flag is set in line 71 (F=1) and the subroutine is returned to the caller. The same comparison takes place in line 305.

Variable U stores the current number of guesses (searches), while variable MX stores the maximum number of guesses (searches) allowed. In other words, if the number of guesses are larger than the maximum number allowed, set the flag and RETURN (LINE 71), or GOTO 420 (line 305).

Just how do we determine what the maximum number of guesses (searches) should be? Some years ago, when I first became interested in binary searches, I knew that there was a relationship between how many pieces of data and the maximum number of searches needed for the worst case scenario. I knew it, I just didn't know what it was. At that time I had a small office, with a computer table and chair, a bookcase, and a regular recliner. I must have gotten up from the computer a hundred times, pacing in circles, throwing myself into the recliner, leaning back, closing my eyes, while trying to find method to the madness. Sometimes I'd end up going to sleep, but most often I would get back up to the computer and start the whole process over.

Finally, the 101st time, after falling asleep in the recliner, I woke up about 2 a.m., and in my sweet dreams I had come up with the answer. Actually, now that I knew the solution, I was surprised that I hadn't thought of it sooner; it was that easy. The maximum number of searches needed to find any number in a given range is:

1 + the power of two of the given range.

The table below should clearly indicate the relationship between the amount of data and the maximum number of searches need to find a particular piece of data.

pieces of data	power of two	max search
1	2 ^ 0	1
2	2 ^ 1	2
4	2 ^ 2	3
8	2 ^ 3	4
16	2 ^ 4	5
32	2 ^ 5	6
64	2 ^ 6	7
128	2 ^ 7	8
256	2 ^ 8	9
512	2 ^ 9	10
1024	2 ^10	11
2048	2 ^11	12
4096	2 ^12	13
8192	2 ^13	14
16384	2 ^14	15
32768	2 ^15	16
65536	2 ^16	17

Note that the maximum number of searches needed changes only at binary numbers. For example, the max number of searches needed to find a particular piece of data in a list of 7, will be the same as in a list of 4. Or, the max number of searches needed to find something in a range of 7371 pieces of data is the same as if there had only been 4096 pieces of data.

To prove this theory, I wrote the following short program in Basic. It is called BINDEMO/BAS and, if you look closely, you'll find that it very much resembles the search routines in GEO4/BAS. You can use the program by setting variable D in line 10 to the amount of data you wish to search, and variable A in line 105 to the particular number you wish to find.

BINDEMO/BAS

```
1 'BINDEMO/BAS
2 'works on Model I, III, 4
3 'binary search demonstration
4 'set variable D in line 10 to the data range and
5 'set variable A in line 105 to the number to search for.
6 'copyright (c) 1993 by Lance Wolstrup
7 'all rights reserved
8 '

10 DEFINT A-Z:D=100:DIM DT(D)

11 FOR X=1 TO D:DT(X)=X:NEXT
12 MX=0:D1=D
13 IF D1/2<1 THEN MX=MX+1:GOTO 100
ELSE D1=INT(D1/2):MX=MX+1:GOTO 13

70 F=0:L=1:R=D:M=INT(D/2):U=1:PRINT U,L,M,R
71 IF U>MX THEN F=1:RETURN
73 IF A=DT(M) THEN RETURN
74 IF A>DT(M) THEN L=M+1 ELSE R=M-1
75 M=INT((R-L)/2)+L:U=U+1
76 PRINT U,L,M,R:GOTO 71

100 CLS
105 A=5
110 GOSUB 70:PRINT:PRINT U;"tries"
120 IF F THEN PRINT"not found"
```

Before we go on to the listing to GEO4/BAS, let me just point out that lines 12 and 13 (in both listings) calculate the maximum number of searches allowed by dividing the number of pieces of data by two. This is repeated until the number is smaller than one. At this point variable MX contains the power of two. We then add one more to MX and exit the routine.

GEO4/BAS

```
1 'GEO4/BAS
2 'an educational game for TRS-80 Model 4
3 '(c) copyright 1993 by Lance Wolstrup
4 ' all rights reserved

10 DEFINT A-R,U-Z:D=435:DIM DT$(D),DT(D):
EN$=CHR$(30)+"<SHIFT><UP-ARROW> to end
game":
NA$="<SHIFT><UP-ARROW> for name prompt":
POKE &H7C,PEEK(&H7C) OR 16
11 PRINT CHR$(15);:
FOR Y=1 TO 3:
FOR X=0 TO 34:
READ A:HD$(Y)=HD$(Y)+CHR$(A):NEXT:
NEXT:CLS:
FOR V=11 TO 13:A$=HD$(V-10):GOSUB 21:NEXT:
FOR Y=1 TO D:READ DT$(Y):DT(Y)=0:NEXT
12 CLS:
FOR V=20 TO 22:A$=HD$(V-19):GOSUB 20:NEXT:
V=20:A$="An educational game for TRS-80 Model 4":
GOSUB 22::V=21:
A$="Copyright (c) 1993 by Lance Wolstrup":GOSUB 23:
V=22:A$="All rights reserved":GOSUB 23:HH=H
13 V=19:A$=STRING$(80,131):GOSUB 20:
POKE &HB94,PEEK(&HB94) OR 12
14 PT$(1)=CHR$(143)+CHR$(244)+CHR$(245)+
CHR$(246):PT$(2)=STRING$(4,32)
15 MX=0:D1-=D
16 IF D1/2<1 THEN MX=MX+1:GOTO 100 ELSE
D1=INT(D1/2):MX=MX+1:GOTO 16

17 DATA 150,190,135,131,167,175,148,32,32,134,147,
   191,131,131,131,133,32,32,150,190,135,131,131,167,
   175,148,32,32,128,152,185,135,170,170,149
18 DATA 149,191,128,128,172,172,148,32,32,128,149,
   191,140,174,128,128,32,32,149,191,128,128,128,170,
   170,149,32,32,134,142,143,140,174,170,157
19 DATA 165,175,180,176,180,190,133,32,32,152,177,
   191,176,176,176,148,32,32,165,175,180,176,176,182,
   190,133,32,32,128,128,128,128,154,186,181

20 H=0:GOTO 23
21 H=INT((80-LEN(A$))/2):GOTO 23
22 H=80-LEN(A$)
23 PRINT@(V,H),A$;:RETURN

30 A$=STRING$(ML,46):GOSUB 23:
A$=CHR$(14):GOSUB 23:FL=0:L=0:A$=""
31 POKE &H74,PEEK(&H74) OR 32:I$=INKEY$:
IF I$="" THEN Z=RND(2):GOTO 31
32 IF I$=CHR$(13) THEN PRINT CHR$(15);:
PRINT@(V,H),CHR$(30);:RETURN ELSE
IF I$=CHR$(27) THEN PRINT CHR$(15);:FL=1:
RETURN
33 IF I$=CHR$(8) AND L=0 THEN 31
34 IF I$=CHR$(8) THEN PRINT CHR$(15);:H=H-1:
PRINT@V*80+H,CHR$(46);:PRINT@V*80+H,"";:
L=L-1:A$=LEFT$(A$,L):PRINT CHR$(14);:GOTO 31
35 IF I$<CHR$(32) THEN 31
36 IF L=ML THEN 31
37 PRINT I$;:H=H+1:A$=A$+I$:L=L+1:GOTO 31

50 S=R*80:T=&HF800+S-1:
IF T<0 THEN T=T+65536!
51 S2=INT(S/256):S1=S-S2*256:
T2=INT(T/256):T1=T-T2*256
52 POKE &HCD7,S1:POKE &HCD8,S2:
POKE &HDA8,S1:POKE &HDA9,S2
```

```
53 POKE &HCC5,T1:POKE &HCC6,T2:
POKE &HD20,T1:POKE &HD21,T2
54 IF T1+1>255 THEN T1=0:T2=T2+1 ELSE
T1=T1+1
55 POKE &HD0B,T1:POKE &HD0C,T2
56 RETURN

60 PRINT@(4,25+LEN(PL$(1))-3),USING"###";PT(1);:
PRINT@(4,51),USING"###";PT(2);:
RETURN

70 F=0:L=1:R=D:M=INT(D/2):U=0
71 IF U>MX THEN F=1:
RETURN
73 IF A$=DT$(M) THEN 78
74 IF A$>DT$(M) THEN L=M+1 ELSE R=M-1
76 M=INT((R-L)/2)+L:U=U+1:GOTO 71
78 IF DT(M)=1 THEN F=2 ELSE DT(M)=1
79 RETURN

100 R=24:GOSUB 50:
V=23:H=HH:A$=EN$:GOSUB 23:
R=19:GOSUB 50:POKE &HB94,PEEK(&HB94) AND 248:
CLS:V=12:A$="Please enter your name: ":GOSUB 20:
H=H+LEN(A$)
101 ML=10:GOSUB 30:NM$=A$:IF FL THEN 120 ELSE
IF A$="" THEN 101
102 PT(1)=0:PT(2)=0:
G=1:PL=0:PL$(1)=NM$:PL$(2)="TRS-80"
103 R=24:GOSUB 50:
V=23:H=HH:A$=NA$:GOSUB 23:
R=19:GOSUB 50
105 V=0:A$="GEO4 SCOREBOARD":GOSUB 21:
V=1:H=25:A$=STRING$(29,140):GOSUB 23:
V=3:A$=PL$(1):GOSUB 23:H=54-LEN(PL$(2)):
A$=PL$(2):GOSUB 23:GOSUB 60:
V=5:A$=STRING$(80,140):GOSUB 20:
POKE &HB94,PEEK(&HB94) OR 6
107 V=17:
A$="Press <ENTER> when ready to begin ":
GOSUB 21:H=H+LEN(A$)
108 ML=1:GOSUB 30
109 IF FL THEN 100 ELSE IF A$<>"" THEN 107
110 V=6:A$=CHR$(31):GOSUB 20
112 R=24:GOSUB 50:V=23:H=HH:A$=EN$:
GOSUB 23:R=19:GOSUB 50
115 FE=0:X=1:RN=RND(D):
A$="Computer randomly selects "+DT$(RN):
PR$=DT$(RN):DT(RN)=1:V=18:GOSUB 20:
PRINT:GOTO 200

120 R=24:GOSUB 50:
POKE &HB94,PEEK(&HB94) AND 240:
POKE &H7C,PEEK(&H7C) AND 239:CLS:END

200 V=17:A$=STRING$(3,13):GOSUB 20:I
F X>2 THEN X=1

210 A$=PL$(X)+" selects: ":GOSUB 20:H=H+LEN(A$)
220 IF PL=0 AND G/2=INT(G/2) THEN 300
230 ML=20:GOSUB 30:IF FL THEN 400 ELSE
IF A$="" THEN 210
235 IF LEFT$(A$,1)<>RIGHT$(PR$,1) THEN
PRINT@(17,0),STRING$(2,13):
A$="Your city or state must begin with "+
RIGHT$(PR$,1):GOSUB 20:GOTO 200
240 GOSUB 70
250 IF F=1 THEN PRINT@(17,0),STRING$(2,13):
A$=A$+" was not found in database - try again":
GOSUB 20:GOTO 200
260 IF F=2 THEN PRINT@(17,0),STRING$(2,13):
A$=A$+" has already been used - try again":
GOSUB 20:GOTO 200
270 PT(X)=PT(X)+1:GOSUB 60:
X=X+1:G=G+1:PR$=DT$(M):GOTO 200

300 F=0:L=1:R=D:M=INT(D/2):U=1
302 A$=RIGHT$(PR$,1)
305 IF U>MX THEN 420
310 IF A$=LEFT$(DT$(M),1) THEN 316
311 IF A$>LEFT$(DT$(M),1) THEN L=M+1
ELSE R=M-1
313 M=INT((R-L)/2)+L:U=U+1:GOTO 305
316 IF M=1 THEN 320 ELSE M=M-1
317 IF A$=LEFT$(DT$(M),1) THEN 316
318 M=M+1
320 IF DT(M)=1 THEN 318
330 IF A$<LEFT$(DT$(M),1) THEN F=2:GOTO 420
340 PR$=DT$(M):DT(M)=1:A$=DT$(M):GOSUB 23
350 PT(X)=PT(X)+1:GOSUB 60:G=G+1:X=X+1:
IF FE THEN 420 ELSE 200

400 FE=1:G=G+1:X=X+1:IF X>2 THEN X=1
410 GOTO 200

420 PRINT@(17,0),STRING$(3,13)
430 R=24:GOSUB 50:
V=23:H=HH:A$=CHR$(30):GOSUB 23:
R=19:GOSUB 50:
V=17:A$="G A M E  O V E R":GOSUB 21
440 PRINT:PRINT:PRINT
450 IF PT(1)=PT(2) THEN A$="** TIE GAME **":
GOTO 500
460 A$=" is the winner"
470 IF PT(1)>PT(2) THEN Y=1 ELSE Y=2
480 A$=PL$(Y)+A$
500 GOSUB 21
510 PRINT:PRINT:PRINT
```

```
520 A$="Would you like to play again (Y/N) ":
GOSUB 21:H=H+LEN(A$)
530 ML=1:GOSUB 30:IF A$="N" OR FL THEN 120
540 IF FL=1 OR A$="N" THEN 120
550 IF A$="Y" THEN FOR Y=1 TO D:DT(Y)=0:NEXT:
CLS:GOTO 102
560 GOTO 520

1000 DATA ABERDEEN,ABILENE,AIKEN,AKRON,
ALABAMA
1001 DATA ALAMAGORDO,ALASKA,ALBANY,
ALBUQUERQUE,ALEXANDRIA
1002 DATA ALLENTOWN,ALTOONA,AMARILLO,
ANAHEIM,ANCHORAGE,ANDERSON
1003 DATA ANN ARBOR,ANNAPOLIS,APPLETON,
ARDMORE,ARIZONA,ARKANSAS
1004 DATA ARLINGTON,ASHEVILLE,ASHLAND,
ASHTABULA,ATCHISON,ATHENS,ATLANTA
1005 DATA AUBURN,AUGUSTA,AURORA,AUSTIN,
BALTIMORE,BANGOR,BATON ROUGE
1006 DATA BEAUMONT,BELLEVUE,BELLINGHAM,
BEND,BERLIN,BETHLEHEM,BILLINGS
1007 DATA BILOXI,BIRMINGHAM,BISMARCK,
BLOOMINGTON,BOISE
1008 DATA BOSTON,BRIDGEPORT,BROOKLYN,
BUFFALO
1009 DATA BURLINGTON,BUTTE,CAIRO,CALIFORNIA,
CAMDEN,CANTON
1010 DATA CAPE GIRARDEAU,CARROLLTON,
CARSON CITY,CASPER,CEDAR RAPIDS
1011 DATA CHARLESTON,CHARLOTTE,
CHATTANOOGA,CHEVY CHASE,CHEYENNE,CHICAGO
1012 DATA CINCINNATI,CLEARWATER,CLEVELAND,
CODY,COLORADO,COLUMBIA
1013 DATA COLUMBUS,CONCORD,CONNECTICUT,
CORPUS CHRISTI,CORVALLIS
1014 DATA COUNCIL BLUFFS,COVINGTON,DALLAS,
DALTON,DANVILLE,DAVENPORT,DAYTON
1015 DATA DECATUR,DELAWARE,DENMARK,DENVER,
DES MOINES,DETROIT
1016 DATA DICKINSON,DODGE CITY,DOTHAN,
DOVER,DUBUQUE,DULUTH
1017 DATA DURANGO,DURHAM,EASTON,
EAU CLAIRE,EFFINGHAM,EL CAJON,EL PASO
1018 DATA ELGIN,ELIZABETH,ELIZABETHTOWN,
ELKHART,ELKO,ELLENSBURG,ELMIRA,ELOY
1019 DATA ELY,ELYRIA,ENCINO,EPHRATA,ERIE,
ESCANABA,ESSEX,EUGENE
1020 DATA EUREKA,EVANSTON,EVANSVILLE,
EVERETT,FAIRBANKS
1021 DATA FAIRFIELD,FARGO,FAYETTEVILLE,
FLAGSTAFF,FLINT
1022 DATA FLORENCE,FLORIDA,FLUSHING,
FORSYTH,FORT LAUDERDALE
1023 DATA FORT SMITH,FORT WAYNE,FORT WORTH,
FRANKFORT,FRESNO,GAINESVILLE
1024 DATA GALVESTON,GARLAND,GARY,GASDEN,
GEORGIA,GETTYSBURG
1025 DATA GILLETTE,GLASGOW,GOLDSBORO,
GRAND FORKS,GRAND ISLAND,GRAND RAPIDS
1026 DATA GREAT FALLS,GREEN BAY,GREENSBORO,
GREENVILLE,GULFPORT
1027 DATA HAMILTON,HAMMOND,HANNIBAL,
HANOVER,HARRISBURG,HARTFORT
1028 DATA HATTIESBURG,HAWAII,HELENA,
HOLLYWOOD,HOMESTEAD,HONOLULU
1029 DATA HOUMA,HOUSTON,HUNTINGTON,
HUNTSVILLE,IDAHO,IDAHO FALLS
1030 DATA INDEPENDENCE,ILLINOIS,INDIANA,
INDIANAPOLIS,INDIO,INGLEWOOD
1031 DATA INVERNESS,IOWA,IRVINE,IRVING,
JACKSON
1032 DATA JACKSONVILLE,JAMESTOWN,
JEFFERSON CITY,JERSEY CITY,JOHNSTOWN
1033 DATA JOPLIN,JUNEAU,KALAMAZOO,KANKAKEE,
KANSAS
1034 DATA KANSAS CITY,KENOSHA,KENTUCKY,
KINGMAN,KINGSTON
1035 DATA KISSIMMEE,KLAMATH FALLS,KNOXVILLE,
KOKOMO,LA JOLLA,LAFAYETTE
1036 DATA LAKELAND,LANCASTER,LANSING,LAREDO,
LAS CRUCES
1037 DATA LAS VEGAS,LEBANON,LEXINGTON,
LIMA,LINCOLN,LISBON,LITTLE ROCK
1038 DATA LONDON,LORAIN,LOS ANGELES,
LOUISIANA,LOUISVILLE,LUBBOCK
1039 DATA LYNCHBURG,MACON,MADISON,MAINE,
MARYLAND
1040 DATA MASSACHUSSETTS,MEDFORD,
MELBOURNE,MEMPHIS,MESA,MESQUITE
1041 DATA METROPOLIS,MIAMI,MICHIGAN,MIDLAND,
MILAN,MILWAUKEE,MINNEAPOLIS
1042 DATA MINNESOTA,MINOT,MISSISSIPPI,
MISSOURI,MOBILE
1043 DATA MONTANA,MONTGOMERY,MONTPELIER,
MORGANTOWN,MUNCIE
1044 DATA MUSKEGON,MUSKOGEE,NAMPA,
NANTUCKET,NAPA,NAPLES,NASHVILLE
1045 DATA NATCHEZ,NEBRASKA,NEVADA,
NEW HAMPSHIRE,NEW HAVEN,NEW JERSEY
1046 DATA NEW MEXICO,NEW ORLEANS,
NEW ROCHELLE,NEW YORK,NEWARK,NEWPORT
1047 DATA NEWPORT NEWS,NIAGARA FALLS,
NILES,NORFOLK,NORMAN,NORTH CAROLINA
1048 DATA NORTH DAKOTA,NORWICH,OAKLAND,
OCALA,ODESSA,OGDEN
1049 DATA OHIO,OKLAHOMA,OKLAHOMA CITY,
OLYMPIA,OMAHA,ONTARIO
1050 DATA OREGON,ORLANDO,OSHKOSH,
```

OWENSBORO,OXFORD,OXNARD
1051 DATA PADUCAH,PALM SPRINGS,PARIS,
PASADENA,PATERSON,PAWTUCKET
1052 DATA PENNSYLVANIA,PENSACOLA,PEORIA,
PHILADELPHIA,PHOENIX
1053 DATA PICAYUNE,PIERRE,PINE BLUFF,
PITTSBURGH,POCATELLO,PONTIAC
1054 DATA PORTLAND,PROVIDENCE,PROVO,RACINE,
RALEIGH,RAPID CITY,READING
1055 DATA RENO,RHODE ISLAND,RICHMOND,
RIVERSIDE,ROANOKE,ROCHESTER
1056 DATA ROCK ISLAND,ROCKFORD,ROLLA,
ROME,SACRAMENTO,SAGINAW
1057 DATA SAINT LOUIS,SAINT PAUL,SALEM,
SALISBURY,SALT LAKE CITY,SAN ANTONIO
1058 DATA SAN DIEGO,SAN FRANCISCO,SAN JOSE,
SANTA BARBARA,SANTA FE,SANTA MONICA
1059 DATA SAVANNAH,SCHENECTADY,SCOTTSDALE,
SCRANTON,SEATTLE,SELMA,SHERIDAN
1060 DATA SHREVEPORT,SIOUX CITY,SIOUX FALLS,
SOUTH BEND,SOUTH CAROLINA
1061 DATA SOUTH DAKOTA,SPARKS,SPOKANE,
SPRINGFIELD,STANFORD,STATEN ISLAND
1062 DATA STEUBENVILLE,STOCKTON,SYRACUSE,
TACOMA,TALLAHASSEE,TAMPA,TEMPE
1063 DATA TENNESSEE,TERRE HAUTE,TEXARKANA,
TEXAS,TITUSVILLE,TOLEDO,TOPEKA
1064 DATA TRINIDAD,TROY,TUCUMCARI,TUCSON,
TULSA,TUPELO,TUSCALOOSA,TWIN FALLS
1065 DATA TYLER,UKIAH,UNION CITY,UPLAND,
URBANA,UTAH,UTICA
1066 DATA VALDOSTA,VALLEJO,VANCOUVER,
VENICE,VENTURA
1067 DATA VERMONT,VERO BEACH,VICKSBURG,
VIRGINIA,VISALIA,WACO,WALLA WALLA
1068 DATA WARWICK,WASHINGTON,WATERBURY,
WATERLOO,WATERTOWN,WAUSAU,WAYCROSS
1069 DATA WEST PALM BEACH,WEST VIRGINIA,
WICHITA,WILMINGTON,WINSTON SALEM
1070 DATA WISCONSIN,WORCESTER,WYOMING,
XAVIER,XENIA,YAKIMA,YANKTON,YAZOO CITY
1071 DATA YONKERS,YORK,YOUNGSTOWN,
YPSILANTI,YREKA,YUKON,YUMA
1072 DATA ZANESVILLE,ZION

Using FASTTERM to access our TRSURETROVE BBS

by Roy T. Beck

Our TRSuretrove BBS is running under Mel Patrick's FASTPLUS II, and very well at that, now that I have learned most of its quirks and vagaries (knock wood!). Actually, considering its size and complexity, it is a marvelous program, and both it and the computer (with hard drive) have been very solid and dependable. Even with the recent Southern California storms, there has been no rebooting of that machine. I had anticipated there might be trouble, but not a bit.

Any user calling into the TRSuretrove BBS must use a terminal program to do so. While there are many programs available, and they do not even have to run on a TRS-80 machine, this article is devoted to Mel Patrick's FASTTERM II, which is explicitly written to operate with the FASTPLUS II BBS program on a Model 4.

Until recently, I did not have the documents for the FASTTERM II program, and so was unable to advise on all of its features. I am still only a neophyte in this regard, but I have been doing some studying and reading, and will try to acquaint you with some of its more arcane features.

First, some info on versions. The original series of FASTTERM programs (without a II on it) went through numerous revisions and iterations. I have, somewhere, a version 10.?. More recently, I have acquired FASTTERM II, version 3.08. An acquaintance in Columbus Ohio has a FASTTERM II, version 4.xx. I don't know for sure what is the latest version, but all of these are workable programs. I strongly suspect that every version has minor bugs in it somewhere, because this is a truism in complex software, becoming ever more true as features are added. Regardless of that, FASTTERM is a very nice program, and I recommend it to you. Please, do contribute the user's fee, currently $10 to Mel Patrick so he will have some incentive to continue upgrading it. Nothing's for free in this world!

Features

There are numerous clever features hidden away "under the hood" of FASTTERM II, and I will only touch lightly upon them in this essay. Most of them work through the hidden menus which are available, usually, with only two keystrokes.

The Buffer

FASTTERM has a buffer for capturing incoming material. The size is 8K if you have a 64K machine, but it goes up to about 70K if you have 128K of RAM. When the buffer is in use, there is a small display near the top of the screen showing the size and the present amount of contents in the buffer. The buffer can be opened and closed under manual control or remote control if the remote computer sends the appropriate commands to control it.

The buffer contents can be viewed at your discretion, and it can be cleared or appended to as required. There is also provision for doing a screen dump to the buffer instead of to the printer.

Finally, there is provision for saving the buffer to a disk file for later review, editing, etc.

Macros via the SCRIPT language

The SCRIPT language allows you to create macros which can perform a whole sequence of actions in very short order. Since Lance Wolstrup is writing a separate article on this feature, which may be found elsewhere in this issue, I'll say no more about it here.

The DIALing menu

Auto-dialing is supported, if your MODEM has the capability, and all the better ones do. To use it, just open the Auto-dial window and push key 1-9 to choose the number you want from your preloaded list. If the line is busy, Automatic redial is the way to go. Another nice feature is that the program will look up your selected area

code and display the local time at that area code. Neat! If your modem and/or PBX, etc require a pause somewhere in the dialing string, the program will accept them at the appropriate point in the string. By the way, the string can include a command to reset the baud rate, if necessary. (Useful if the board you are calling has only a low speed MODEM at its end).

The Autodial window also includes (if you enable it) an elapsed time display so you can always tell how long you have been on line, and further, a $ display showing what your accumulated charge is for the call at any time. You have to load in the rate per minute to make this feature work, but it's there.

The Library command menu

This feature allows you to execute any DOS LIBRARY command without exiting FASTTERM. Need to know how much FREE space is on a drive? Need to do a DIR call? This is the feature for suchlike needs.

The File transfer menu

This is the means to upload and download files. It can handle straight ASCII files with the XON/XOFF protocol, and any files via XMODEM and YMODEM. YMODEM is similar to XMODEM, except it uses 1024 byte blocks instead of the 128 byte blocks in XMODEM. The larger blocks allows faster transfer with good, clean phone lines, but may actually slow you down on noisy lines, as larger blocks must be resent in case of error due to line noise. There is also a free space map to allow you to determine where you are going to save a download. Very handy.

Printer Control

FASTTERM allows you to set some printer parameters, and also includes a small buffer between itself and your printer.

Terminal Emulation

FASTTERM has limited ability to emulate other terminals. Besides it's default mode, which is essentially that of the teletype, it can emulate two others. One is VIDTEX, used when signed onto COMPUSERVE. This mode is deliberately crippled to prevent COMPUSERVE controlling some things in your program which Mel Patrick thinks COMPUSERVE should not be able to control! The other mode is VT100, which emulates a popular terminal unit. The emulation allows you to do some screen editing. Since I don't know or use the VT100, I won't attempt to explain the effects. If you use and like the VT100, then this will make FASTTERM II act very much like a VT100.

Personalization

FASTTERM has a control character translation table which you are free to edit if you need to. It is 31 characters in length, and will translate 01H to 1FH to something else, if you need it. There are a few other items which may be altered in FASTTERM, such as the name of the FT/CFG file.

Splitting the screen

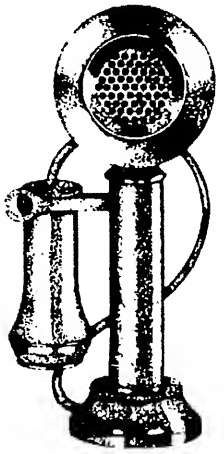
There are times when it is desirable to split the screen into two parts, one for incoming text and one for outgoing text. FASTTERM allows this type of operation. Two lines near the top of the screen are reserved for your outgoing text; the remainder of the screen shows what is incoming. In this mode, you can type up to 159 characters without actually sending them, and then send them in a burst by hitting the ENTER key. Meanwhile, the other party is sending material to you, and the messages are not intermixed. One situation where this would be useful is in the CHAT mode with the SYSOP, or in the CB mode on COMPUSERVE. CTL V toggles the split screen mode on or off.

Altogether, FASTTERM is a very slick and elegant terminal program, and if you don't have it, or don't have the latest version, why not drop a letter to Mel Patrick (with $10 in it), and he will return the latest disk and docs to you. Try it, you'll like it!

Mel's address is:
Mel Patrick
8056 164A St.
Surrey, British Columbia V3S 7S7
604-574-2072 (BBS)

HOW I FINALLY GOT ON THE PHONE

by Jim E. King,
Electrical & General Contractor



Since I use the Model 4P in my business, and I primarily like to write and collect applications software, I have always had plenty to interest and occupy my time on the computer without dialing up bulletin boards. However, when Roy & Lance whopped up their new TRSuretrove BBS, I decided that now was the time.

Lance said that he bought a modem at a local computer swap meet for $52 for Roy, and I got the model # and the name of the manufacturer: 2400X, by Best Data Products, Inc., Chatsworth, CA (818/773-9600, 9304 Deering Av., 91311). I wanted exactly the same modem because it had been proven to work. I phoned them, and they wouldn't sell to me!!

After some confusion they recommended that I purchase the modem from one of their dealers, Computerman located in Reseda, CA. Computerman didn't have the 2400X, but did have the 2400XMNP with more bells & whistles for $96.34. I said 'what the hell' and bought it.

It immediately became obvious that I needed more stuff to put the system together. I didn't think that a cable for the PC would work on the TRS-80, so I went to Tandy and bought a 6 foot RS-232C cable, Male DB25 to Male DB25, with TRS molded in the plastic, that works with my 4P. I also bought a telephone 1 outlet to 2 outlet converter.

I wanted a telephone cord about 1 foot long to connect to a telephone, and RS didn't have one, so I went to Sandy's Electronics looking for it. I found that they come in lengths of about 7 inches, and 7 feet, so I bought some plugs and a crimper to make my own. I also found that their phone converters were about $3.50.

Costs:

2400X MNP Modem, Best Data Prod, Chats.	$ 96.34
TRS 6 ft. Modem RS-232 cable, Cat#26-249	$ 16.95
TRS Telephone converter, duplex	$ 4.49
Sandys 7 ft. Telefone cord	$ 3.77
" 11 Telefone plugs	$ .99
" Telefone crimper	$ 9.65

	$132.19

When I got home I found that the modem came with a 7 foot phone cord, and that I already had a converter that I had forgotten. When I find a telefone with a miniature jack in the back I will make a short cord to go to the modem.

The Tandy modem cable has screws on each side of the plug to fasten the connector down so that it can't be pulled out, but they are too short to reach the sockets and are useless. I believe you can get it a lot cheaper somewhere else, like Sandy's.

I finally called TRSuretrove New Year's night, and said: "Help, I need instructions on how to use the FastTerm program." Two people answered my request with some information, and I thank them very much. Roy noted on the BBS that he would try to get documentation from the author.

Now to summarize what I know about the system:

Roy Beck's BBS, TRSuretrove: 213/664-5056
Protocol 8,N,1
Board B is for Model 4.
Board C for Models 1 & 3
Change your password in the X Submenu.

Hayes Modem Commands (MUST USE CAPS):

ATDT 1-213/664-5056 <E>	dials Roy Beck's TRSuretrove BBS
ATA	Answer mode
ATD	Originate mode
AT?	Prints out modem info
A/	Repeat last Cmd
+++	Exit from modem; hang up
REC	Receive
TRS	Request to Send
DTR	Data Terminal Ready

MODEM LIGHTS MEAN:

CD	Carrier Detect
RI	Ring Indicator
OH	Off Hook
RD	Receive Data
SD	Send Data
HS	High Speed
TR	Terminal Ready
MR	Modem Ready
AA	Auto Answer

The following is a list of the FASTTERM II commands. The list has been useful to me, and I am presenting it here so that it will hopefully be a benefit to other new users.

FastTerm II COMMANDS
from Chapter 11 of Version 3.XX
Copyright 1990 by Mel Patrick
Compiled by Jim King

<B>	=	BREAK,
<C>	=	CLEAR
<E>	=	ENTER
<S>	=	SHIFT
<CTL>	=	CTRL

TERMINAL MODE

<B>	Exit to DOS. FastTerm will ask twice.
<CTL>B	Send a true modem break of 350 ms.
F1	Invoke command mode menu selection.
F2	Linefeed filter, only needed if you require linefeeds. If you get double spacing on all received text, try setting the CR + LF to ON.
F3	Toggle 'bell' sound on/off.
<S>F1	Invoke the terminal emulation selection window.
<S>F2	Output a linefeed each time you press <E>. This is normally used with systems which use a separate linefeed and Carriage Return.
<S>F3	Toggle the Custom Selection Window. Provides a menu of all these commands, and disables the <C>letter commands.
<S><C>	Press if the screen becomes unreadable, to restore it to power up condition.
UpArrow	Invoke the video buffer scroll mode. Then you may use the DownArrow to scroll in the opposite direction. Press <B> or <E> to return.
<C>#	# = 1 to 9. Load & execute the Script Filename that is referenced by the # in the Script menu.
<C>C	Store screen display in Capture buffer.
<C>F	Toggle Cost Charges
<C>G	Toggle Capture Buffer OPEN/CLOSED.
<C>H	Fast disconnedt for modems that support DTR signal.
<C>K	Toggle elapsed timer display.
<C>O	Output cost display on screen to log file.
<C>P	Toggle printer ON/OFF.
<C>Q	Abort printer spooling and empty spooler buffer
<C>R	Reset charges or elapsed timer to zero.
<C>S	Suspend charges toggle; blinks a $ to show suspended.
<C>V	Toggle split screen mode
<C>X	Append incomming text to Capture buffer.
<C>Z	Toggle cursor ON/OFF.

References:

Bulletin Boards	TRSTimes 1.2, p21, 3/88
Tim's PD Express	FastTerm, Sewell, TRSTimes 2.1 p18, 1/89
SAGATUG Interface,	12/92
TRSuretrove	TRSTimes 6.1, p5, 1/93
Tips on Modem Use	Banks, TRSTimes 6.1, p10, 1/93
FastTerm II User Manual	Mel Patrick, Chapter 11, p29

SCRIPTING WITH FastTerm II

by Lance Wolstrup

Going to a Valley TRS-80 Hackers' User Group meeting is still, after all these years, one of my favorite computer activities. Why? Because my friends, the people who struggled along with me to get the early hardware and software working, are there.

Sure, the group has shrunk. While we would regularly have 50-100 people show up at the meetings in the early and middle eighties, we now only have the hard-core TRS-80 nuts left.

Also, while we used to spend hours talking about Assembly Language and/or Basic code, the meetings now contain a variety of subject; Dr. Allen Jacobs will talk about just about any subject from his proposed 'mouse interface' to nuclear physics; Jim King will discuss the latest version of his personal database, or Multidos in general; Roy Beck will put together a hard drive from scratch, or perform some other super hardware task, such as diagnose and then fix whatever is wrong with the other members' machines - on the spot.

Since Roy opened the TRSuretrove BBS, we have used the meetings to discuss the various things that pertain to BBSing, and to the TRSuretrove BBS in general. Other than FastPlus, the program that runs the BBS, we have discussed Mel Patrick's FastTerm II, the best and most obvious communication program to use on the Model 4. Though my copy of FastTerm II is a legal, registered copy, I just never had the documentation. Roy contacted Mel Patrick, but still, no docs. Finally, Mike Lingo of the Orange County TRS-80 Users Group came to the rescue and very graciously mailed me a copy of the documentation. Thanks, Mike.

Reading through the newly acquired material, I realized that there was a lot more to FastTerm II than I had first expected. I knew that Mel Patrick was (and still is) an excellent programmer and that this program is one of his best efforts, but I was taken by surprise when I read that FastTerm II has a built-in script language. Holy Moley, only the PC terminal programs have script languages!! Well, obviously not so. Mel has implemented a complete programming language inside of FastTerm II; just about everything you'd need to make your on-line session easier and better has been included.

Being a programmer at heart, I started to play with the script language. Unfortunately, unlike regular languages in which you can test your program directly on your own machine, I needed a BBS to check out the validity of my code. Of couse, I did the obvious, I called the TRSuretrove BBS each time another routine was added. By failing many times, I finally began to understand how the language worked. Over a period of a couple of days, I know that I drove the Beck household crazy, calling in 5 minute increments for several hours. I called so many times, in fact, that I racked up a long distance phone bill of $80 just to the BBS. For those readers not familiar with the local geography, Roy lives in Los Angeles while I live in the western-most part of the San Fernando Valley, the city of Woodland Hills. Now, Woodland Hills is still considered part of greater Los Angeles, but Roy and I live a 'fur piece' apart. We have different area codes, and a call from one to the other ends up being as expensive as calling St. Louis. Yes, programming is a costly undertaking!

Having learned something new, it was just natural that I would present the information at the next 'Hackers' meeting. I did that at the February meeting, and to round out the FastTerm II trilogy in this issue, the following is the pertinent portion of that presentation.

FastTerm II has a programming language (called a script language) built right in to the program. The language consists of approximately 63 commands, as well as modem commands and data, each designed to automate a large portion of an on-line session.

Like any other programming language, the FastTerm scripts have commands that are used more than others. We will concentrate on a few of the more common (and most useful) ones, and in the process create a script that will automatically call the TRSuretrove BBS, display an elased timer, answer the name and password prompts and leave the user right at the main menu. Nothing very fancy, but very useful. It is a script that I now use each time I call.

First, a script is an ASCII file containing the instruction you wish FastTerm II to carry out. It can be written on any word processor or text editor capable of writing ASCII files. My personal preference is TED. It is available on all LS-DOS 6.3.1 system disks; it is a very capable editor, and it is fast - it is perfect for the task.

Second, all FastTerm script commands begin with a special symbol. This symbol is the left bracket: [
On a TRS-80 Model 4, the left bracket is generated by pressing <CLEAR><,>

Third, the script that we are about to write assumes that you, the user, is already a caller to the board. That is, it assumes that you already have a password. If this is not the case, log on to the board manually and go through the initial first-time caller sequence, asking you

to choose a password. The next time you call, the script will completely automate the log-on and place you in the main menu.

Fourth, when keying in the script, do NOT type the comments following each command (in assembly language style). The comments are placed there to explain what the command does and is strictly for the readers' eyes; FastTerm will choke on them.

Finally, commands to the modem assume that you are using the industry standard, a Hayes (or compatible). If not, you will have to translate to your modem commands, or be out of luck - I have enough trouble with the Hayes command set!

Now, to get started, invoke TED and we are ready to write our script.

ATDT 1 213 664-5056	dial trsuretrove bbs.
[WAIT FOR CONNECT	continue script when the 'connect' message appears.
[CLOCK RESET	set clock to 00:00:00.
[CLOCK ON	display clock.
[WAIT FOR Name	continue script when 'name' prompt appears.
YOUR NAME HERE	insert your name on this line.
[WAIT FOR Correct	continue script when 'correct' prompt appears.
Y	yes, name is correct.
[WAIT FOR Password	continue script when 'password' prompt appears.
YOUR PASSWORD HERE	insert your password here.
[WAIT FOR continue	continue script when 'continue' prompt appears.
[CTRL M	simulate pressing the enter key in response to prompt.
[END	end script.

As I said earlier, type only the bolded lines, pressing <ENTER> after each line, and make sure that you replace the name and password entries with your own.

The script is now done, so save it to disk by pressing <CTRL><F>. Call the file TRSURE/SCR. Then exit TED.

If TRSURE/SCR is not on the same disk as FastTerm II, copy it over there now. It really is not necessary, rather it is for good measure.

Now, fire up FastTerm II. When the program is loaded, get into the Scripts menu by pressing <CLEAR><M>.

(As an aside, <CLEAR><M> is used to invoke the Scripts menu because in a previous version this used to be the 'Macro' menu. I presume that Mel simply forgot to change the keystroke code!)

Inside the Scripts menu, press <E> and then type 1 in response to the # prompt.

This highlights the dots at the 1> entry below.

At this point type the name of the script: TRSURE/SCR and press <ENTER>.

Finally, save the script entry by pressing <S>. This should place you back in terminal mode.

Now, whenever you wish to execute this script, press <CLEAR><1> directly from terminal mode.

This was just a very simple script. Many different, much more complicated things can be accomplished using Mel Patrick's Script language. For those of you wishing to jump in with both feet we present a compilation of Chapter 15 of the FastTerm II manual, listing the script commands.

SCRIPT COMMAND LIST

Parameters are shown in italics after the command. Note that some commands may use variables instead of the options parameters shown in italics.

[APPEND	If the caprure buffer has been used, you may use this command to append text to what is currently in the buffer without having to clear it out first. If the buffer has not been used, this will work the same as the CAPTURE ON command.
[BAUD *rate,WPS*	where rate can be 300, 1200, 2400, or 9600. W=word, P=parity, and S=stop bits. (WPS is usually 8N1)
[BEEP	will produce a BEEP from the Model 4 internal sound board.
[BREAK	will send a true break of 350 ms to the modem. In some cases this may cause a disconnect.
[BUFFSAVE *filespec/ext:d*	will save anything in the capture buffer, providing there is actually something in the buffer. If not, the command is ignored.
[CAPTURE *on/off*	will open or close the capture buffer, depending on the ON or OFF word. If no word is present, ON is assumed.
[CHAIN filespec/ext:d	file must exist or command will fail.
[CHARGE AUTO on/off	will control output of the charges on loss of carrier. If no argument then ON is assumed.

[CHARGE COST H/M### will set the cost of a call in H for hours or M for minutes to the number entered. Do not use periods to specify a decimal value. $1.42 per hour would be entered as H142.

[CHARGE *on/off* will control the display of the charges time cost on the screen. If no argument is given, ON is assumed.

[CHARGE RESET resets the cost charges timer to 00:00:00

[CHARGE RESUME will re-enable the charges to add up again. The $ on the screen will stop flashing to indicate that charge timing has resumed.

[CHARGE SUSPEND will suspend cost charges from adding up untill CHARGE RESUME is used. The $ on the screen will flash to indicate suspension.

[CHARGE WRITE if CHARGE is ON, this will write the total charges to the disk and turn off the charge display on the screen.

[CLOCK *on/off* will controll the elapsed time as per the command word. If no argument is supplied, ON is assumed.

[CLOCK RESET will reset the elapsed timer to 00:00:00

[CLS will clear the screen

[COPY %# %# can be used when you want to copy from one variable to another. [COPY %1 %2 is an acceptable format.

[CRLF on/off FastTerm II has the ability to handle carriage returns and linefeed characters separately. If you have this option on, a CR will move the cursor to the start of the line and a LF will advance it one line. With it off, a LF character will be ignored and a CR will move the cursor to the start of the line and automatically generate a linefeed.

[DEFAULT this will reset the following items (whether they are presently on or off): trace mode, remote echo, local echo, clock display, outgoing linefeeds, emulation set to TTY, no auto output of charges, charge display, printer output, capture buffer, and split screen.

[DTR will cause the RS232 to drop DTR and will resume with the proper level approximately 2 seconds later. This will almost always enable the modem to terminate the connection.

[ECHO *on/off* will control the local echo, as per the ON or OFF word. If no argument is specified, ON is assumed.

[ECHOR *on/off* will turn the remote echo on or off as indicated by the ON or OFF word. If no argument is specified, ON is assumed

[END this is an optional command. Do try to get into the habit of using it to end your scripts.

[ESC causes FastTerm II to send an ESCAPE code to the remote. The esc value is 1BH or 27 decimal. This value will not be shown on your Model 4 screen unless sent back from the remote.

[FLUSH will completely empty the RS232 buffer of any received characters in it.

[GET1KX *filespec/ext:d* will receive a file from the remote using the filename specified. It uses the 1K xmodem CRC protocol.

[GETLINE *(#) V#* where (#) is the length of the input up to 80 chrs and V# is a variable from 0 to 4. This command will allow you to receive directly into one of the 5 variables allowed in FastTerm II up to 80 chrs (default, or whatever you have it set to). Storing in the variable will end at the first control character, carriage return linefeed, or when the specified length is received, and the script will the continue to process.

[GETXMD *filedpec/ext:d* will receive a file from the remote using the filename specified. Uses standard xmodem protocol with auto CRC/Checksum detection.

[GOSUB *label* where label is defined elsewhere in the script as a label. The subroutine must always end with the [RETURN statement. You may nest GOSUBs up to 10 levels deep.

[GOTO *label* where the label is defined elsewhere in your script as a label. The goto is an unconditional move to a new location. You may use a variable to pass the name of the label you want to move to. Once there, the script processor will start to execute the code immediately following the *label* command.

[IF {*str1*} {=?} {*str2*} where str1 is the string you want to check for, and str2 is the string you want to check it against. Options are = where both strings must match to length of str1. Second option is ? which will check to see if the string in str1 is contained in str2. All IF statements must end with [ENDIF. Failure to end conditional IF statements with [ENDIF will cancel the script operation.

[KEYIN "*text*" *V#* will display a prompt on the screen so that you may define one of the variables directly from the keyboard. The # sign may be a number from 0 to 4.

[LABEL *label name* where label name is a name was referred to by either a GOTO or GOSUB command. FastTerm II will take no action of any kind when reading or stepping through the script file when it comes across a label command. They are only used for the GOTO and GOSUB routines. Only alphanumeric characters are allowed in a label name.

[LINEFEEDS *on/off* will control the outgoing linefeeds. if no argument is supplied, ON is assumed.

[MASKBIT *on/off* will enable/disable the 8th bit received as is done in the Buffer window menu.

[PAUSE will immediately stop execution of the script file. The [S] at the top of the screen to indicate scripting is in effect will flash to indicate pause mode. To remove the pause and continue with the script operation, use the key combination of <CLEAR><C>.

[PRINT *on/off* routes incoming text to the printer, depending on the ON/OFF switch.

[REDIAL phone number will redial the number following this command. You should have a HAYES modem and be using the AT command set. FastTerm II will continue to redial the number until either a connection is made OR you press the <SPACE BAR> to cancel redial and advance the script.

[REM anything appearing on this line will be ignored. It serves only as a programming aid when writing scripts.

[RESET TC will reset both elapsed time and charge time to 00:00:00

[RETURN this command must always be used to terminate a GOSUB routine.

[SEND "*text*" *or %1* this command will send out either a literal string enclosed in quotes or a variable.

[SET *V#* will define the variable referenced by the # sign from within the script. Only 39 characters may be used, and any additional ones will be purged. The # sign may be any number from 0 to 4.

[SND1KX *filespec/ext:d* will send an xmodem file to the remote using the 1K xmodem CRC protocol.

[SNDASC *filespec/ext:d* will send an ASCII file to the remote using the filename specified. Note that it will not be shown on your video display as it is sent.

[SNDXMD *filespec/ext:d* will send a standard xmodem file to the remote using the filename specified. Will automatically detect if you need CRC or Checksum modes

[SPLIT *on/off* split screen mode will be enabled or disabled. If no argument, ON is assumed.

[SWAP %# %# this command will swap two variables. The # signs can be any variable numbers from 0 to 4. The information in each variable will be exchanged with the other.

[TRACE *on/off* this command causes the script processor to display 45 chrs from each of the script commands read in from the script file. The top line of the video is used for display. Each time the command is displayed, there will be a two second delay so that the operator has time to see it.

[TTY will select NO emulation of any kind.

[TWAIT ## where the ## is the delay in seconds you want to wait. No characters can be sent or displayed on your screen during this mode, and scripting is paused until the time specified is completed.

[VIDTEX will cause FastTerm II to react to the Compuserve VIDTEX screen formatting.

[VT100 will cause FastTerm II to simulate a VT100 terminal.

[WAIT FOR "*text*" will copy the string following this command to a special buffer and will pause scripting until the string is matched. The string may be a word, sentence or partial form of either so long as the total length is under 41 characters.

[WAIT TIME *hh:mm:ss* will cause FastTerm II to wait until the time in the script matches the time on your Model 4 clock. It provides an exact match, so ensure that a correct time is used. The clock string must be in the format HH:MM:SS and in 24 hour mode. If you want to continue processing while in a WAIT TIME command, just press the <SPACE BAR>. FastTerm II will cancel the time matching and continue operation.

[WINCLR erase all contents of the FastTerm II script display window. This is the window you fill with text using the [WINLINE command.

[WINDSP display current window (or whatever is in it at the time).

[WINHIDE remove current window from screen. (hide it).

[WINKEY "*text*" *V#* does an input inside the displayed window. It only works properly if the window is active, and there's no way to tell if the window is not present. Uses the bottom line of the window as an input line and doesn't disrupt the video display. Usually 15 characters are available.

[WINLINE # "text" where # is the line number in the window, and text is what you want to move there. It is always padded remove any previous characters. A null simply erases that line. There are 6 lines for your use in this command (1-6) which replace the # sign. Up to 58 characters will be used from your literal text or variable.

[WRITE "*text*" will display the string of text following on the screen, but will not send it to the modem.

SPECIAL CONTROL CHARACTERS

There are some special control characters which can be entered from the keyboard while in the scripting mode which will affect the operation of the script. Press and hold down <CLEAR> and then <SHIFT> and finally the letter of the function.

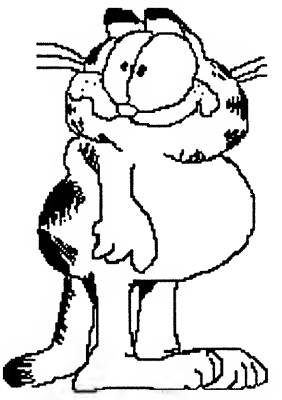
CLEAR SHIFT A will abort the script currently being used. There is no recovery from this command.

CLEAR SHIFT C if the script is in [WAIT FOR mode and for some reason misses the string, press this combination and the script will continue as if the string was received. This key combination is also used to start the script again after a [PAUSE command.

CLEAR SHIFT E if your script has an error in it, such as a command not recognized, use this command to bring the error line to the display.

CLEAR SHIFT V will display the current version of the scripting command processor.

CLEAR SHIFT W will display the total command list currently supported by the scripting command processor.



HINTS & TIPS

SCROLDEM/BAS REVISITED

by Henry H. Herrdegen

Typing in SCROLDEM/BAS, I couldn't help myself to jazz it up a bit. Below is my version of the program with the changes bolded.

```
1 ' scroldem/bas
2 ' (c) 1992 by Lance Wolstrup
3 ' all rights reserved
4 ' nevertheless jazzed up by H3
10 GOTO 100
60 IF R>23 THEN R=0
61 CB=&HF800+R*80:
IF CB<0 THEN CB=CB+65536!
62 CBH=INT(CB/256):CBL=CB-CBH*256
63 POKE &HB95,CBL:POKE &HB96,CBH:
POKE &HC06,CBL:POKE &HC07,CBH:
POKE &HC24,CBL:POKE &HC25,CBH:
POKE &HCD4,CBL:POKE &HCD5,CBH:
POKE &HD75,CBL:POKE &HD76,CBH
64 CS=1920-R*80:CSH=INT(CS/256):
CSL=CS-CSH*256
65 POKE &HCD7,CSL:POKE &HCD8,CSH:
POKE &HDA8,CSL:POKE &HDA9,CSH
66 RETURN
100 CLS:PRINT"Scroll protectong 9 lines. . ."
110 FOR X=1 TO 15:PRINT X+1 STRING$(76,191);:
NEXT
120 R=9:GOSUB 60:PRINT@(16,0),;
130 IF INKEY$="" THEN PRINT STRING$(80,42)
STRING$(80,45);:GOTO 130
140 PRINT@0,"Now scroll protecting 2 lines ";
150 R=2:GOSUB 60
155 PRINT@(23,0),;
160 IF INKEY$="" THEN PRINT STRING$(80,35)
STRING$(80,45);:GOTO 160
170 PRINT@0,"Scroll protecting 15 lines . . ."
180 R=15:GOSUB 60:
PRINT@(14,0),"this is the 15th line. " CHR$(31)
190 PRINT@(23,0),;
200 IF INKEY$="" THEN PRINT STRING$(80,140)
STRING$(80,61);:GOTO 200
210 PRINT@0,"Scroll protecting 22 lines . . ."
220 R=22:GOSUB 60:PRINT@(21,0),"here is line 22: "
230 PRINT@(23,0),;
240 IF INKEY$="" THEN FOR X=1 TO 80:
PRINT CHR$(94);:NEXT ELSE 250
242 FOR X=1 TO 80:PRINT CHR$(124);:NEXT:
GOTO 240
250 R=0:GOSUB 60
260 CLS:FOR X=0 TO 22:
PRINT STRING$(80,191);:NEXT:PRINT"hit key to clear";
270 IF INKEY$="" THEN 270
280 FOR R=23 TO 0 STEP-1:GOSUB 60:CLS:END
```

FIND THAT CURSOR

Model 4/LS-DOS 6.x.

by Lance Wolstrup

If you program in Model 4 Assembly Language, you probably know that function 4 of the @VDCTL supervisor call returns the current position of the cursor. The value in register H is the cursor's vertical position, while the value in register L reflects the cursor's horizontal position.

Model 4 Basic has two commands that will tell you where the cursor is located. POS(0) stores the horizontal position in the variable assigned to the expression; for example, H=POS(0).

ROW(0) stores the vertical position of the cursor in the variable assigned to the expression; for example, V=ROW(0).

So, both languages provide a means to find the cursor, but - just where do the languages find this information? Where is the cursor position stored?

B95H and B96H form the address of the cursor position - but wait - it is not quite as easy as that; the values found at B95H and B96H form the location of the cursor offset from F800H. Let me demonstrate what I mean:

First, we need to turn the values in B95H and B96H into a 16-bit value. We do this by multiplying the value stored at B96H by 256, and then adding the value found at B95H. Now, subtract 63488 (F800H) from this 16-bit value, and we now have the PRINT@ cursor position. If we wish to break down this value further, to the actual vertical and horizontal position, we do the following:

Divide the newly-found PRINT@ position by 80, and retain only the integer portion; this is the vertical value. Multiply the vertical value by 80, and subtract the result from the original PRINT@ value; this is the horizontal value.

If we were going to do this in Basic, it would look something like this:

```
10 HL=(PEEK(&HB96)*256+PEEK(&HB95))-63488!
20 V=INT(HL/80)
30 H=HL-V*80
```

To demonstrate the above Basic program, add line 5 and line 40, and then RUN the program.

```
5 PRINT@(20,18),""; 'cursor at vertical 20, horizontal 18
```

```
40 PRINT"Vertical =";V;"  Horizontal =";H
```

This initially positions the cursor at 20,18. After manipulating the values found at B95H and B96H, as explained above, line 40 confirms that the vertical position of the cursor is 20, while the horizontal position is 18.

Now we now where that little sucker is stored!

DISABLE THE BREAK KEY
MODEL I/III LDOS
by Lance Wolstrup

LDOS, like its big brother LS-DOS, uses one bit in a particular memory location to determine the functionality of the Break key.

Setting bit 4 of 442BH diables the Break key, while resetting this bit enables the Break key. Here is how to do it from Basic.

Disable the Break key:
```
POKE &H442B,PEEK(&H442B) OR 16
```

Enable the Break key:
```
POKE &H442B,PEEK(&H442B) AND 239
```

FIND THAT CURSOR
Model I/III - all DOSes
by Lance Wolstrup

On the previous page, I talked about how to find the current cursor position while programming the Model 4. I mentioned that Model 4 Basic have the POS(0) and ROW(0) commands. That is just fine for the Model 4. Unfortunately, these commands do not exist for the Models I and III. Therefore we have to resort to pretty much the same technique we presented as the altenate method in that tip; that is, we have to manually figure out where the current cursor is located.

Anyone programming in Model I/III Assembly Language is undoubtably familiar with where the position of the cursor is stored in memory - 4020H (16416 decimal) and 4021H (16417 decimal).

In order to find the cursor position from within Basic, we have to extract the Most Significant Byte from 4021H and the Least Significant Byte from 4020H. This is done as follows:

```
10 HL=(PEEK(&H4021)*256+PEEK(&H4020))-15360
```

At this point variable HL contains the PRINT@ position of the cursor; that is, we could: PRINT@HL,"Whatever".

If we wanted to further break down the cursor position into vertical and horizontal information, we could add this code:

```
20 V=INT(HL/64)
30 H=HL-V*64
```

Now you also know where the cursor is on the Model I and III.

LASER DEMO

by Frank Gottschalk

Original was printed in color as indicated:

TRS-80 WINS AGAIN ←Red

COLOR PRINTING AVAILABLE
FROM MODEL 4'S WITH A ←green
H P DeskJet printer.

Standard TRS-80 Model 4's can now print in color on a Hewlet Packard DeskJet printer with Laser sharp quality. The software used for this bulletin was the Allwrite word processor and the DeskJet Utility Pack. – blue ¶

This versatile combination allows the use of a multitude of font *styles* as well as sizes. As a matter of fact, there is a program available to convert any **LaserJet** font to a **DeskJet** font, making your selection limitless. These can then be downloaded into the DeskJet RAM Memory cartridge for use, and will stay there until you reset the printer. – black ¶

The DeskJet provides the LaserJet quality printing at 300 dpi. Albeit, not as fast, but considering the loading time for the Laser before it starts printing, it's not too bad. Considering the price differential (paramount for me), although not as great as it used to be, it's an excellent trade off for the money. Of course there is a draft mode for faster printing of test copies. – brown ¶

It's the colored ink refill packs now available for the DeskJet cartridges that make – red the color possible. Just refill used cartridges with the colored ink to get the colors. – green For multi-colored printing however requires a separate pass for each color, or – blue pausing the printer to change the cartridge color. Perhaps a nuisance, but at least it – brown works. And the cost is reasonable.

Model 4
Public Domain Disks

M4GOODIES#1: day/cmd, day/txt, gomuku/cmd, llife/cmd, llife/doc, writer/cmd, writer/doc, writer/hlp, yahtzee/bas

M4GOODIES#2: arc4/cmd, arc4/doc, cia/bas, etimer/cmd, index/cmd, index/dat, mail/bas, mail/txt, trscat/cmd, trscat/txt, util4/cmd, xt4/cmd, xt4/dat, xt4hlp/dat

M4GOODIES#3: convbase/bas, dates/bas, dctdsp/cmd, dmu/cmd, dmu/doc, dskcat5/cmd, dskcat5/doc, editor/cmd, editor/doc, fedit/cmd, fkey/asm, fkey/cmd, fkey/doc, hangman/cmd, m/cmd, m/src, membrane/bas, miniop2/cmd, miniop2/src, move/cmd, move/doc, othello4/bas, scroll4/cmd, scroll4/src, setdate6/cmd, setdate6/doc, setdate6/fix, spaceadv/bas, taxman/bas, utilbill/bas, utilbill/doc

M4GOODIES#4: WORD WIZARD disk #1 of 3
anima/bas, archi/bas, autos/bas, battuere/bas, captus/bas, contvert/bas, curro/bas, dico/bas, ducere/bas, eulogos/bas, facere/bas, fluere/bas, gradi/bas, jacere/bas, kata/bas, male/bas, metron/bas, naus/bas, startup/bas, startup/jcl, stig/bas, tangere/bas, wordmenu/bas

M4GOODIES#5: WORD WIZARD disk #2 of 3
cognos/bas, frangere/bas, juris/bas, medius/bas, mittere/bas, monos/bas, numbers/bas, orare/bas, pandemos/bas, para/bas, pathos/bas, pendere/bas, philanth/bas, phongrap/bas, polynom/bas, prefix1/bas, prefix2/bas, premere/bas, sal/bas, startup/bas, startup/jcl, statuere/bas, wordmenu/bas

M4GOODIES#6: WORD WIZARD disk 3 of 3
bible/bas, french1/bas, french2/bas, french3/bas, italian/bas, latphras/bas, lit1/bas, lit2/bas, myths/bas, places/bas, plicare/bas, spanish/bas, stagnare/bas, stare/bas, startup/bas, startup/jcl, synpath/bas, televid/bas, tenere/bas, vaco/bas, valere/bas, vox/bas, wordmenu/bas

M4GOODIES#7: calendar/cmd, castladv/bas, civilwar/bas, crimeadv/bas, dctdsp/cmd, ed6/cmd, ed6/doc, edittext/bas, fedit/cmd, mail/bas, mail/txt, scramble/bas, states/bas, textpro/cmd, time4/bas, wizard/bas, wizard/doc, worldcap/bas

M4GOODIES#8: books/bas, books/doc, dmu/cmd, dmu/doc, hamcalc/bas, hamhelp/bas, network/bas, network/doc, pirate/bas, pirate/doc, vmap/bas, vmap/doc, vmap2/bas, vmap2/doc, zork1/doc, zork2/doc, zork3/doc

M4GOODIES#9: ft/cmd, ft/doc, pterm/cmd, pterm/doc, r/cmd, r/doc, scrconv/bas, scrconv/doc, video4/asm, video4/cmd

M4GOODIES#10: checker/cmd, crossref/cmd, crossref/doc, ddir/cmd, diskcat/cmd, diskcat/doc, division/bas, division/doc, getput/bas, getput/doc, host/cmd, hv/bas, maszap4/cmd, maszap4/doc, park/cmd, profile4/doc, protect/bas, protect/doc, rename/bas, replace/bas, restore/bas, rm/bas, scrndump/bas, scrndump/doc, super/hlp, vers/cmd

M4GOODIES#11: benchmrk/bas, bigcal/bas, bigcal/doc, birthday/bas, dearc4/cmd, dezip2/cmd, dname/cmd, docufile/bas, docufile/doc, docufile/mrg, escape/bas, mem4/cmd, million/bas, nomad/bas, password/bas, password/dat, password/doc, password/jcl, roman/bas, sixtymin/bas, startrek/bas, trekinst/bas

M4GOODIES#12: awari/bas, buyimg/bas, crasher/bas, curvfit2/bas, gradebk/bas, mortcost/bas, mortcost/doc, print/bas, print/doc, reiman/bas, square/bas, starlane/bas, staybus/bas, sunrise/bas, synonym/bas, timezon1/bas, timezon2/bas, travel/bas, vmap2/bas, vmap2/doc, weekday/bas

M4GOODIES#13: calndr1/bas, calndr2/bas, calndr3/bas, formltrs/bas, invloan/bas, limerick/bas, martian/bas, mission/bas, moneymkt/bas, munchmth/bas, numbrfun/bas, smith/bas, smith/doc, star2000/bas, starfind/bas, starfind/dat, starfind/doc, starfind/jcl, states/bas, wallst/bas

M4GOODIES#14: alphahex/bas, bowlchng/bas, bowlcrea/bas, bowldetl/bas, bowlfinl/bas, bowling/doc, bowlmenu/bas, bowlprnt/bas, bowlrcap/bas, bowlrecd/bas, bowlrecp/bas, bowlschd/bas, bowlscor/bas, bowlsort/bas, buscheck/bas, calculat/bas, chekform/bas, deprec/bas, futrdate/bas, membrain/bas, minimath/bas, normalz/bas, numconv/bas, pcbdest/bas, pcbdest/doc, pcform/bas, pcpm/bas, pcpm/doc, pcpm/jcl, utscan/bas, yagibeam/bas, zeller/bas

M4GOODIES#15: laughs/bas, laughs/dat, laughs/doc, laughs1/dat, laughs2/dat, laughs3/dat, laughs4/dat, laughs5/dat, laughs6/dat, laughs7/dat, laughs8/dat, laughs9/dat, laughs10/dat, laughs11/dat, laughs12/dat, laughs13/dat, laughs14/dat, laughs15/dat

M4GOODIES#16: trivia/bas, trivia/doc, trivia1/dat, trivia2/dat, trivia3/dat, trivia4/dat

M4GOODIES#17: acrs/bas, amorloan/bas, clokmod/bas, compound/bas, dcform/bas, decide/bas, easyword/bas, editno/bas, epslabel/bas, esckey/bas, expect/bas, funct1/bas, funct2/bas, gasform/bas, hexprint/bas, hexsay/bas, lostgold/bas, mathfunc/bas, mpgcalc/bas, neclabel/bas, nicelist/bas, nonlin/bas, nonlin/rem, payback/bas, peekprnt/bas, percent/bas, prntcall/bas, proverbs/bas, randseed/bas, savings/bas, speech/bas, tasklist/bas, tempconv/bas, weightfm/bas

Each disk is $5.00 (U.S.)
or get any 3 disks for $12.00 (U.S.)
please specify the exact disks wanted.

TRSTimes PD-DISKS
5721 Topanga Canyon Blvd., Suite 4
Woodland Hills, CA. 91367

PROGRAMMING TIDBITS

BASIC IMPS

Raise your hands, please: how many have used the basic imp? Do I see a hand back there? Oh, you never heard of a basic imp? You know, a small devil, one of the six "logical operators" in Mod-4 BASIC, used mainly in IF commands to "bind" two or more comparisons into a "true" or "false" expression. If you are like the majority of BASIC programmers (including yours truly) you probably rarely use the IMP. Most of the time we do just fine with AND, OR, plus possibly NOT, the trio inherited from the Mod-I days.

Mod-4 BASIC introduced three more imps, courtesy of Microsoft: XOR, EQV and the IMP himself. Though not essential, they can often make our logic more "elegant". For a somewhat trivial example suppose we're computing a function of two variables. The function is valid only when both variables are positive or zero, or both are negative. We could write:

```
IF (x>=0 AND y>=0) OR (x<0 AND y<0) then...
```

This kind of test can be greatly simplified with the EQV operator which stands for "EQuiValent". It's a sort of rigid all-or-nothing position: "either both must be true, or both must be false". No half-truths are tolerated, yet our test is cut by half:

```
IF x>=0 EQV y>=0 then...
```

When both comparisons are true then the expression is true, like before. Miraculously, when both comparisons are false (X is negative and Y is negative) then the combined expression is also true, which is what we need in this case. Not quite common sense but so are many things in computers. The EQV reminds me of the old advice: "tell the whole truth, but if you must lie then lie all the way" (supposedly one can even cheat the lie detector this way, though I never had the chance to try).

The XOR operator performs a service similar to EQV, but its logic is opposite. The mnemonic stands for "eXclusive OR" and says: "either one must be true, or the other, but not both". The difference between a "normal" OR and XOR is that the former is "true" when one term is true, or the other, or both. Using XOR our test could be written:

```
IF x>=0 XOR y<0 then...
```

Here, when X>=0 is true (X positive), then Y<0 must be false (ie. Y must be also positive). Or the other way around. If both comparisons are true, or both are false, then the test fails, which is the exact opposite of the EQV logic.

The IMP proper is the weirdest of all imps. The mnemonic stands for "IMPlies". Not only it's confusing but also unique, because it's the only logical operator where the meaning and order of terms are significant. Its conversational approximation is a sentence such as "it's raining, the street is wet". If we reverse the order of the terms ("the street is wet, it's raining") then the expression is meaningless, because a wet street does not necessarily imply that it must be raining (maybe it's a dry, hot summer day and kids are fooling around with a hydrant).

The peculiar consequence of this logic is that an IMP test always returns "true" except when the first term is true and the second false. In scorecard games there is usually a sensible rule that you can't lose points if your total is already below minimum, until you again earn some points (banks seem to work the same way: they won't lend you money unless you already have money). If S is your score and T is your total so far, then we could write:

```
IF s<0 IMP t>min then t=t+s
```

A losing hit (S<0 is true) requires (implies) that your total be above minimum (T>min must be also true), otherwise it doesn't count. When a hit is zero or positive (S<0 is false) then the expression will be always true. If you're not losing, we don't ask what your total is (your friendly banker doesn't care how much money you keep in your account, as long as you're not writing checks on "insufficient funds").

As noted, the order of IMP terms is significant. If we'd mix up the terms and wrote:

```
IF t>min IMP s<0 ...
```

...then the logic would not make any sense, because the rules of our game do not require that you must lose points when your total is above minimum. With all other logical operators a reversal of terms makes no difference.

In a way IMP is the opposite of AND which is always "false" except when both terms are true. So our test could be also written:

```
IF s<0 AND t<=min then goto... else t=t+s
```

This more closely resembles the way our rule was stated in English, and we'd probably let it go. But from

the programming point of view the IMP would be more direct, even if a bit more tricky.

Since any logical expression can be constructed with one or more "common-sense" imps AND, OR, NOT, we usually stick with the tried and true. But those trusty oldtimers can be quite impish, too. Take the AND, for example:

```
IF k$ > = "A" AND k$ < = "Z"
```

...is always true when K$ is an uppercase letter. Not much can go wrong here, or can it? Many a programmer (including yours truly) spent hours debugging an AND code that didn't produce the expected results. It has to do with the difference between the strictly logical language of computers, and the somewhat sloppy human way of talking. A typical case: I want to beat the mail gridlock and send early holiday cards to friends in "France and Italy". My carefully crafted maillist program accordingly asks:

```
IF country$ = "France" AND country$ = "Italy"...
```

...then print the envelope. What the imp? Nothing comes out, though I am sure I typed in at least two dozen names. Panic! Data files crashed? Hours and hours of typing wasted? No. It's just that no one lives in France AND Italy at once. The proper way to ask the computer would be:

```
IF country$ = "France" OR country$ = "Italy"...
```

Let's hope computers will never replace waiters. If you'd ask for oil AND vinegar on your salad, the computer would give you no dressing at all. If you'd suggest French OR Italian dressing, the computer would cheerfully serve a mixture of both.

The NOT operator can be even more treacherous. On the face of it, it simply produces the opposite of a condition. For example in my mailing program:

```
IF country$ = "USA" then...
```

...will return addresses of my American friends, while:

```
IF NOT country$ = "USA" then...
```

...will correctly produce addresses which are NOT in this country. The troubles start when we get clever. Internally BASIC takes a comparison and generates a number 1 when true, 0 if false. In binary notation 1 is 11111111. The NOT flips each "one" to "zero" and the other way around. Thus "NOT-true" correctly produces "false" (00000000). However, to simplify the internal BASIC code, its designers decided that not only 1, but any value other than zero will mean "true". One side benefit of this approach is that it allows convenient programming shortcuts. For example to search for a string K$ in an array we might write something like this:

```
for x = 1 to 1000
IF instr( a$(x),k$ ) > 0 then ...
next
```

Since any non-zero value means "true", we can just as well omit the ">0" and write:

```
IF instr( a$(x),k$ ) then ...
```

This not only saves a few bytes, but actually can speed up a program, especially when this kind of operation is repeated many times, maybe in a long loop. Not a bad deal, as long as we keep in mind what it really means. But in the last minute rush to "improve" the code, a clever programmer (including yours truly) remembers from kindergarten days that "no" means simply "no", quickly sticks a NOT into a program line, and hopes that the result will be exactly the opposite of the original (in our example "true" only for array elements where string K$ is NOT found):

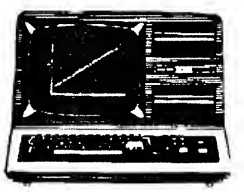
```
IF NOT instr( a$(x),k$ ) then ...
```

The effect is not even close to the intent. Our clever revision has made the test totally useless. Now it's "true" in all cases. Consider: INSTR returns either 0 (string not found) or some positive value (position of the search string in the array element). Any value other than 1, has at least one zero in its binary form. The NOT flips each zero into a "one" and generates some non-zero value which means "true" again. Thus NOT is always true, unless the operand is "truly" true to begin with: a variable or expression with value 1, or a "true" result of a comparison.

In the official manuals (but not in those "by Chris") the rules for the imps are usually given in "truth tables" full of zeros and ones, or T's and F's. I always had trouble with those impish tables, until I realized that the "truth" of each imp boils down to one plain English statement:

OR	true when at least one true
XOR	true when only one true
AND	true when both true
EQV	true when both true or both false
IMP	true when both true or first false
NOT	true unless operand "truly" true (-1)

So now you know what imps are. Can I see those hands again? How many used the IMP?



HUNTING FOR BURIED TREASURE
PEEKING & POKING MODEL 4
LS-DOS 6.x.x - Basic
by Lance Wolstrup

This month's installment was inspired by a letter received from Henry Herrdegen, who writes:

A program I was working on recently got me to look again into scroll protecting more than the regulation 7 lines, so I dug up all you and David Goben had written about the subject. I never could get Dave's patch from CN80 to work neither in 6.3.0 nor 6.3.1, even with the corrections published later. Why???
Henry H. Herrdegen
LaSalle, Ontario, Canada

I have to admit that I was not familiar with Dave Goben's scroll protect patches. Fortunately, Henry included a printout of them in his letter After looking through the code I will have to defend Dave; he is a good programmer and his code usually works. These patches are no exception. They do work as promised.

First, let's discuss exactly what the patches do, and then, when we understand what is going on, let's see how we can accomplish the exact same thing from Basic, without resorting to the patch method.

Patch #1.
D0A,CE = 3A 9E 0D E6 0F
F0A,CE = DD 7E 00 E6 07

The patch begins at memory location 0CCEH where the code is as follows:

```
0CCE    DD7E00          LD   A,(IX+0)
0CD1    E607            AND  7
```

Register IX points to memory location 0B94H, so the contents of 0B94H is loaded in to register A. Since only the lowest three bits (0-2) control the scroll protect, the top five bits (3-7) are masked out with the AND 7 instruction. Register A now holds the value of the scroll protect.

This code is being replaced by:

```
0CCE    3A 9E 0D        LD   A,(0D9EH)
0CD1    E6 0F           AND  15
```

Rather than getting the scroll protect value from 0B94H where only the lowest 3 bits are available, we now get it from memory location 0D9EH. We then mask out the top 4 bits with the AND 15 instruction, thus leaving only the scroll protect value in register A.

Patch #2.
D0A,F5 = 0F 4F 3A 9E 0D
F0A,F5 = 07 4F 3A 94 0B

Patch #2 begins at memory location 0CF5H. For the sake of clarity, let's move back one byte to 0CF4H; there we find the instruction: AND 7. Since we are not going to replace the AND instruction, but only the data, the patch begins at the data location, which is 0CF5H.

The actual assembly language instruction found, beginning at 0CF4H are:

```
0CF4    E607            AND  7
0CF6    4F              LD   C,A
0CF7    3A940B          LD   A,(0B94H)
```

This routine has at this point the user desired scroll protect value in register A. It is then ANDed with 7 to force modulo 8. This new value is the copied into register C, and register A is then loaded with the value from 0B94H, where the first three bits contain the current scroll protect value.

The patch changes the above code to:

```
0CF4    E615            AND  15
0CF6    4F              LD   C,A
0CF7    3A9E0D          LD   A,(0D9EH)
```

The routine now forces modula 16 for the user desired scroll protect. The new value is copied into register C, and register A is loaded with the value from location 0D9EH, which contain the current scroll protect value.

Patch #3.
D0A,FA = E6 F0 B1 32 9E 0D
F0A,FA = E6 F8 B1 32 94 0B

This is simply a continuation of where the code left off at patch #2.

```
0CFA    E6F8            AND  0F8H
0CFC    B1              OR   C
0CFD    32940B          LD   (0B94H),A
```

By ANDing register A with F8H, we remove the current scroll protection, then, by OR C, we merge in the user desired number of lines to protect. Finally, this new value is stored in location 0B94H.

The patch modifies the above code to read:

```
0CFA    E6F0            AND  F0H
0CFC    B1              OR C
0CFD    329E0D          LD   (0D9EH),A
```

Since we are now working with modulo 16 (instead of modulo 8), we remove the current scroll protection by ANDing the current value with F0H (240). We then merge

the user desired scroll protect (OR C) and store the new value in memory location D9EH.

Patch #4.
D0B,9B = AF 18 0C 00
F0B,9B = ED B0 AF C9

The actual code is:

```
0D9B    EDB0        LDIR
0D9D    AF          XOR A
0D9E    C9          RET
```

It really doesn't matter what this code does. Rather, the idea is to shorten the code, to free up one byte to be used as the new data location for the scroll protect value. As you will see by the patchcode, the byte that is freed up is 0D9E.

The patch code is:

```
0D9B    AF          XOR A
0D9C    180C        JR   0DAAH
0D9E    00          NOP
```

The code sets register A to 0 (XOR A) as did the original routine, and then jumps 12 bytes further up in the code to 0DAAH where we find the code LDIR, followed by CP A, and RET. Location 0D9EH is now capable of being used for data storage.

The logic behind the patches is good. The scroll protect scheme suffers from the same stinginess of bit allocations as the date in TRSDOS 6.2. The date scheme used 3 bits to store the number of the year, which were then added to 80. Thus, the highest year that could be entered was 1987. Remember that?

The scroll protect scheme uses bits 0-2 of memory location B94H. That is also 3 bits. Thus, the highest value that can be stored there is 7. We cannot use more bits in B94H because the remaining ones are used for tab/spec chrs settings.

But we need more bits in order to expand the amount of lines to scroll protect. Therefore it is logical to try to find another location to store that information. This is not an easy task, since Roy Soltoff writes very compact code, but David found a routine that could be shortened by 1 byte, and this is what is happening in patch #4. The memory location that was freed up by compacting the code is D9EH.

Since we now have a new memory location to store the scroll protect data, we need to redirect all scroll protect references from B94H to D9EH. This is done in patch #1, #2, and #3.

Finally, in order to protect B94H from accepting a scroll protect value higher than 7, and thus blow up the rest of the data in that location, the scroll protect value is being ANDed with 7. This forces the value to be 0 to 7. Since D9EH does not have to worry about destroying bit 4, we can now expand the scroll protect value to be up to 15. That is why patch #1 and 2 changes 07H to 0FH.

Now, my question is this. Why would I want to PATCH my system diskette permanently to gain this new ability when it is far safer to simply use it as a subroutine in a Basic or ML program. Yes, now that I have revealed the actual memory locations used, we can POKE the new values to them

The subroutine would look like this:

```
70 POKE &HCCE,58:POKE &HCCF,158:
POKE &HCD0,13:POKE &HCD2,15
71 POKE &HCF5,15:POKE &HCF8,158:
POKE &HCF9,13
72 POKE &HCFB,240:POKE &HCFE,158:
POKE &HCFF,13
73 POKE &HD9B,175:POKE &HD9C,24:
POKE &HD9D,12:POKE &HD9E,0
74 RETURN

75 POKE &HCCE,221:POKE &H126:
POKE &HCD0,0:POKE &HCD2,7
76 POKE &HCF5,7:POKE &HCF8,148:
POKE &HCF9,11
77 POKE &HCFB,248:POKE &HCFE,148:
POKE &HCFF,11
78 POKE &HD9B,237:POKE &HD9C,176:
POKE &HD9D,175:POKE &HD9E,201
79 RETURN
```

The routine in lines 70 to 75 sets up the ability to scroll protect up to 15 lines from the top. When the calling program get ready to scroll protect, simply POKE &HD9E with the desired number of lines.

Before exiting the program, make sure to GOSUB 75. This routine resets the DOS machine code back to normal.

Add the following lines to the above subroutines and we have a working program to demonstrate the new scroll protection abilities.

```
10 GOTO 100

100 CLS
110 GOSUB 70:POKE &HD9E,15
120 PRINT@(23,0),"";
130 FOR X=1 TO 50:
PRINT"Scroll protecting 15 lines":NEXT
140 IF INKEY$="" THEN 140
150 POKE &HD9E,0
160 FOR X=1 TO 50:
PRINT"Scroll protecting 0 lines":NEXT
170 GOSUB 75
```

In conclusion let me just say that I didn't really mean to take up two pages answering a letter, but the subject was interesting!

Until next time, keep POKEing your Model 4.

ATTENTION TRSDOS 1.3. USERS!

ANNOUNCING "SYSTEM 1.5.", THE MOST COMPREHENSIVE 1.3. UPGRADE EVER OFFERED!

MORE SPEED!! MORE POWER!! MORE PUNCH!!

While maintaining 100% compatibility to TRSDOS 1.3., this DOS upgrade advances TRSDOS 1.3. into the 90's!
SYSTEM 1.5. supports 16k-32k bank data storage and 4MGHZ clock speed (4/4P/4D).
DOUBLE SIDED DRIVES ARE NOW 100% UTILIZED! (all models).

CONFIG=Y/N	CREATES CONFIG BOOT UP	FILEDATE=Y/N	DATE BOOT UP PROMPT ON or OFF
TIME=Y/N	TIME BOOT UP PROMPT ON or OFF	CURSOR='XX'	DEFINE BOOT UP CURSOR CHAR
BLINK=Y/N	SET CURSOR BOOT UP DEFAULT	CAPS=Y/N	SET KEY CAPS BOOT UP DEFAULT
LINE='XX'	SET *PR LINES BOOT UP DEFAULT	WP=d.Y/N (WP)	WRITE PROTECT ANY or ALL DRIVES
ALIVE=Y/N	GRAPHIC MONITOR ON or OFF	TRACE=Y/N	TURN SP MONITOR ON or OFF
TRON=Y/N	ADD an IMPROVED TRON	MEMORY=Y/N	BASIC FREE MEMORY DISPLAY MONITOR
TYPE=B/H/Y/N	HIGH/BANK TYPE AHEAD ON or OFF	FAST	4 MGHZ SPEED (MODEL 4'S)
SLOW	2 MGHZ SPEED (MODEL III'S)	BASIC2	ENTER ROM BASIC (NON-DISK)
CPY (parm,parm)	COPY/LIST/CAT LDOS TYPE DISKS	SYSRES=H/B/'XX'	MOVE/SYS OVERLAY(s) TO HI/BANK MEM
SYSRES=Y/N	DISABLE/ENABLE SYSRES OPTION	MACRO	DEFINE ANY KEY TO MACRO
SPOOL=H/B.SIZE	SPOOL is HIGH or BANK MEMORY	SPOOL=D.SIZE='XX'	LINK MEM SPOOLING TO DISK FILE
SPOOL=N	TEMPORARILY DISABLE SPOOLER	SPOOL=Y	REACTIVATE DISABLED SPOOLER
SPOOL=RESET	RESET (NIL) SPOOL BUFFER	SPOOL=OPEN	OPENS, REACTIVATES DISK SPOOLING
SPOOL=CLOSE	CLOSES SPOOL DISK FILE	FILTER *PR.ADLF=Y/N	ADD LINE FEEDS BEFORE PRINTING 0DH
FILTER *PR.IGLF	IGNORES 'EXTRA' LINE FEEDS	FILTER *PR.HARD=Y/N	SEND 0CH to PRINTER (FASTEST TOF)
FILTER *PR.FILTER	ADDS 256 BYTE PRINTER FILTER	FILTER *PR.ORIG	TRANSLATE PRINTER BYTE TO CHNG
FILTER *PR.FIND	TRANSLATE PRINTER BYTE TO CHNG	FILTER *PR.RESET	RESET PRINTER FILTER TABLE
FILTER *PR.LINES	DEFINE NUMBER LINES PER PAGE	FILTER *PR.WIDTH	DEFINE PRINTER LINE WIDTH
FILTER *PR.TMARG	ADDS TOP MARGIN to PRINTOUTS	FILTER *PR.BMARG	ADDS BOTTOM MARGIN to PRINTOUT
FILTER *PR.PAGE	NUMBER PAGES, SET PAGE NUMBER	FILTER *PR.ROUTE	SETS PRINTER ROUTING ON or OFF
FILTER *PR.TOF	MOVES PAPER TO TOP OF FORM	FILTER *PR.NEWPG	SET DCB LINE COUNT TO 1
FILTER *KI.ECHO	ECHO KEYS to the PRINTER	FILTER *KI.MACRO	TURN MACRO KEYS ON or OFF
ATTRIB:d.PASSWORD	CHANGE MASTER PASSWORD	DEVICE	DISPLAYS CURRENT CONFIG INFO

All parms above are installed using the new LIBRARY command SYSTEM (parm,parm). Other new LIB options include DBSIDE (enables double sided drive by treating the "other side" as a new independent drive, drives 0-7 supported) and SWAP (swap drive code table #s). Dump (CONFIG) all current high and/or bank memory data/routines and other current config to a disk data file. If your type ahead is active, you can (optional) store text in the type buffer, which is saved. During a boot, the config file is loaded back into high/bank memory and interrupts are recognized. After executing any active auto command, any stored type ahead data will be output. FANTASTIC! Convert your QWERTY keyboard to a DVORAK! Route printer output to the screen or your RS-232. Macro any key,even F1, F2 or F3. Load *01-*15 overlay(s) into high/bank memory for a memory only DOS! Enter data faster with the 256 byte type ahead option. Run 4MGHZ error free as clock, disk I/O routines are properly corrected! Spool printing to high/bank memory. Link spooling to disk (spooling updates DCB upon entering storage). Install up to 4 different debugging monitors. Print MS-DOS text files, ignoring those unwanted line feeds. Copy, Lprint, List or CATalog DOSPLUS, LS-DOS, LDOS or TRSDOS 6.x.x. files and disks. Add top/bottom margins and/or page numbers to your hard copy. Rename/Redate disks. Use special printer codes eg: LPRINT CHR$(3); toggles printer output to the ROUTE device. Special keyboard codes add even more versatility. This upgrade improves date file stamping MM/DD/YY instead of just MM/YY. Adds optional verify on/off formatting, enables users to examine *01-*15, DIR, and BOOT sectors using DEBUG, and corrects all known TRSDOS 1.3. DOS errors. Upgrade includes LIBDVR, a /CMD driver that enables LIBRARY commands, such as DIR, COPY, DEBUG, FREE, PURGE, or even small /CMD programs to be used within a running Basic program, without variable or data loss.

By special arrangement with GRL Software,
SYSTEM 1.5. is now distributed exclusively by TRSTimes magazine.
ORDER YOUR COPY TODAY!
Send $39.95 (U.S. funds) to:
TRSTimes - SYSTEM 1.5.
5721 Topanga Canyon Blvd., Suite 4
Woodland Hills, CA. 91367

GRAPHICS-90
NOW for Model I or III

The definitive graphics program that Radio Shack SHOULD have released is now, through the courtesy of author Larry Payne and the hard work of Gary Shanafelt, available from TRSTimes. Our agreement is to sell this wonderful package at COST. Neither Larry Payne, Gary Shanafelt nor TRSTimes will make a dime on this venture. The price charged will cover the cost of reproducing the manual, disks, and mailing only. The only one profiting is YOU!!

GRAPHICS-90 **Model III $ 11.50**
Model I $ 13.00

Shipping: Europe: add $4.50 air mail or $2.75 surface mail
Asia, Austrailia & New Zealand: add $5.25 air mail or $3.50 surface mail
Please specify Model I or III and DOS preference

TRSTimes magazine - dept. GR90
5721 Topanga Canyon Blvd. #4
Woodland Hills, CA 91367

RECREATIONAL & EDUCATIONAL COMPUTING

REC is the only publication devoted to the playful interaction of computers and 'mathemagic' - from digital delights to strange attractors, from special number classes to computer graphics and fractals. Edited and published by computer columnist and math professor Dr. Michael W. Ecker, REC features programs, challenges, puzzles, program teasers, art, editorial, humor, and much more, all laser printed. REC supports many computer brands as it has done since inception Jan. 1986. Back issues are available.

To subscribe for one year of 8 issues, send $27 US or $36 outside North America to REC, Att: Dr. M. Ecker, 909 Violet Terrace, Clarks Summit, PA 18411, USA or send $10 ($13 non-US) for 3 sample issues, creditable.

TRSTimes on DISK #10

Issue #10 of TRSTimes on DISK is now available, featuring the programs from the Jul/Aug,Sep/Oct & Nov/Dec 1992 issues:

TRSTimes on DISK is reasonably priced:

U.S. & Canada: $5.00 (U.S.)
Other countries: $7.00 (U.S.)
Send check or money order to:

TRSTimes on DISK
5721 Topanga Canyon Blvd. #4
Woodland Hills, CA 91367

TRSTimes on DISK #1, 2, 3, 4, 5, 6, 7, 8, & 9 are still available at the above prices

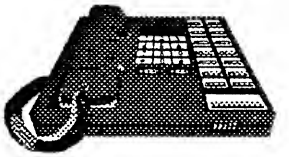
ITEMS FOR SALE

FRESH SUPPLY
of Used Surplus Parts, Pieces, Systems.
Repairs/Upgrades at low prices, all guaranteed.
Models I, II, III, 4, 12, 16, 2000, CoCo.
(MS-DOS too)
From Disks ($.20) to Manuals,
Printers, Plotters, Hard Drives.

Call/Write with offer:
RECYCLER, Frank Gottschalk
785 Maya Ct.
Fremont, CA 94539
(510) 651-2313 8 AM - 11 PM PT)

TRSuretrove BBs

8 N 1 - 24 hours
Los Angeles
213 664-5056



where the TRS-80 crowd meets

MISOSYS, Inc.

CLOSEOUT SALE ON SOFTWARE: NO RETURNS; ALL SALES FINAL

AFM: Auto File Manager data base (3)	$10.00 D	LED / LS-LED (3,4)	$5.00
BackRest for hard drives (3&4)	$10.00	LS-Host/Term (4)	$10.00
BSORT (3,4)	$5.00	LS-UTILITY (4)	$10.00
Comsoft Group 5-Game Disk (3)	$20.00	MC / PRO-MC (3,4)	$79.95 D
CP/M (MM) Hard Disk Drivers	$10.00 B	Mister ED (4)	$10.00 B
CON80Z (3,4)	$5.00	MRAS / PRO-MRAS (3,4)	$30.00 D
dlskDISK (3,4)	$10.00	PowerDot (Epson or Tandy) (3)	$5.00
DoubleDuty (4)	$25.00	PowerDraw (3)	$5.00
DSM51 / DSM4 (3,4)	$10.00	PowerDriver Plus (Epson) (3)	$5.00
DSMBLR / PRO-DUCE (3,4)	$10.00	PowerMail Plus (3,4)	$15.00 D
EDAS / PRO-CREATE (3,4)	$10.00 D	PowerMail Plus TextMerge (3,4)	$5.00
Filters: Combined I & II (3)	$5.00 B	PowerScript (3,4)	$10.00 B
GO:Maintenance (4)	$15.00 B	PRO-WAM (4)	$50.00 D
GO:System Enhancement (4)	$15.00 B	PRO-WAM Toolkit (4)	$15.00 B
GO:Utility (4)	$15.00 B	QuizMaster (3)	$5.00
Hardware Interface Kit (4)	$5.00	RATFOR (4)	$10.00 D
HartFORTH (3,4)	$10.00 B	SuperUtlilityPlus (3,4)	$15.00 D
Kim Watt's Game Hits (3)	$10.00	Supreme HD Driver (3&4)	$15.00
Lair of the Dragon (3&4)	$10.00	TBA / LS-TBA (3,4)	$5.00 D
Lance Miklus' Game Hits (3)	$15.00	The Gobbling Box (3&4)	$10.00
LDOS 5.3.1 Mod1 Upgrade klt (1)	$20.00 B	THE SOURCE 3-Volume Set	$10.00 D
LDOS 5.3.1 Mod3 Upgrade Kit (3)	$20.00 B	Toolbox/Toolbelt (3,4)	$10.00 B
Leo Cristopherson's Game Disk (3)	$10.00	UNREL-T80 (3&4)	$5.00
LS-DOS 6.3.1 Upgrade Kit (4)	$20.00 B	UTILITY-I (3)	$5.00
LS-DOS 6.3.1 Upgrade kit (2)	$25.00 B	XLR8er Software Interface Kit (4)	$5.00 B

MISOSYS, Inc. Sterling, VA 20167-0239
P.O. Box 239 703-450-4181

The Fine Print: Numbers in () indicate computer support - please specify TRS-80 Model. Freight codes: A = $3.50; B = $4.00; C = $4.50; D = $5.00; All unmarked are $3.00 each; Canada/Mexico add $1 per order; Foreign use US rates times 3 for air shipment. Virginia residents add 4.5% sales tax. We accept MasterCard and VISA; Checks must be drawn on a US bank. COD's are cash, money order, or certified check; add $5 for COD.

*** DR. PATCH ***

BRAND-NEW UTILITY FOR TRS-80 MODEL 4 AND LS-DOS 6.3.1

A 'MUST HAVE' FOR ALL LS-DOS 6.3.1 OWNERS.

DR. PATCH MODIFIES LS-DOS 6.3.1 TO DO THINGS THAT WERE NEVER BEFORE POSSIBLE.
COMPLETELY SELF-CONTAINED - MENU-DRIVEN FOR MAXIMUM USER CONVENIENCE.
FAST & SAFE - EACH MODIFICATION IS EASILY REVERSED TO NORMAL DOS OPERATION.

DISABLE PASSWORD CHECK IN FORMAT/CMD
FORMAT DOUBLE-SIDED AS DEFAULT
FORMAT 80 TRACKS AS DEFAULT
DISABLE VERIFY AFTER FORMAT
CHANGE 'DIR' TO 'D'
CHANGE 'CAT' TO 'C'
VIEW DIR/CAT WITH (I) PARAMETER AS DEFAULT
VIEW DIR/CAT WITH (S,I) PARAMETERS AS DEFAULT
CHANGE 'REMOVE' TO 'DEL'
CHANGE 'RENAME' TO 'REN'
CHANGE 'MEMORY' TO 'MEM'
CHANGE 'DEVICE' TO 'DEV'
DISABLE THE BOOT 'DATE' PROMPT
DISABLE THE BOOT 'TIME' PROMPT
DISABLE FILE PASSWORD PROTECTION
ENABLE EXTENDED ERROR MESSAGES

DISABLE PASSWORD CHECK IN BACKUP/CMD
BACKUP WITH (I) PARAMETER AS DEFAULT
BACKUP WITH VERIFY DISABLED
DISABLE BACKUP 'LIMIT' PROTECTION
DISABLE PASSWORD CHECK IN PURGE
PURGE WITH (I) PARAMETER AS DEFAULT
PURGE WITH (S,I) PARAMETERS AS DEFAULT
PURGE WITH (Q=N) PARAMETER AS DEFAULT
IMPLEMENT THE DOS 'KILL' COMMAND
CHANGE DOS PROMPT TO CUSTOM PROMPT
TURN 'AUTO BREAK DISABLE' OFF
TURN 'SYSGEN' MESSAGE OFF
BOOT WITH NON-BLINKING CURSOR
BOOT WITH CUSTOM CURSOR
BOOT WITH CLOCK ON
BOOT WITH FAST KEY-REPEAT

DR. PATCH IS THE ONLY PROGRAM OF ITS TYPE EVER WRITTEN FOR THE TRS-80 MODEL 4 AND LS-DOS 6.3.1. IT IS DISTRIBUTED EXCLUSIVELY BY TRSTIMES MAGAZINE ON A STANDARD LS-DOS 6.3.1 DATA DISKETTE, ALONG WITH WRITTEN DOCUMENTATION.

*** DR. PATCH *** $14.95

SHIPPING & HANDLING: U.S & CANADA - NONE
ELSEWHERE - ADD $4.00
(U.S CURRENCY ONLY, PLEASE)

TRSTimes magazine - dept. DP
5721 Topanga Canyon Blvd. #4
Woodland Hills, CA 91367

DON'T LET YOUR LS-DOS 6.3.1 BE WITHOUT IT!